Università di Trieste

Task Manager

Enrico Piccin - Corso di Basi di Dati

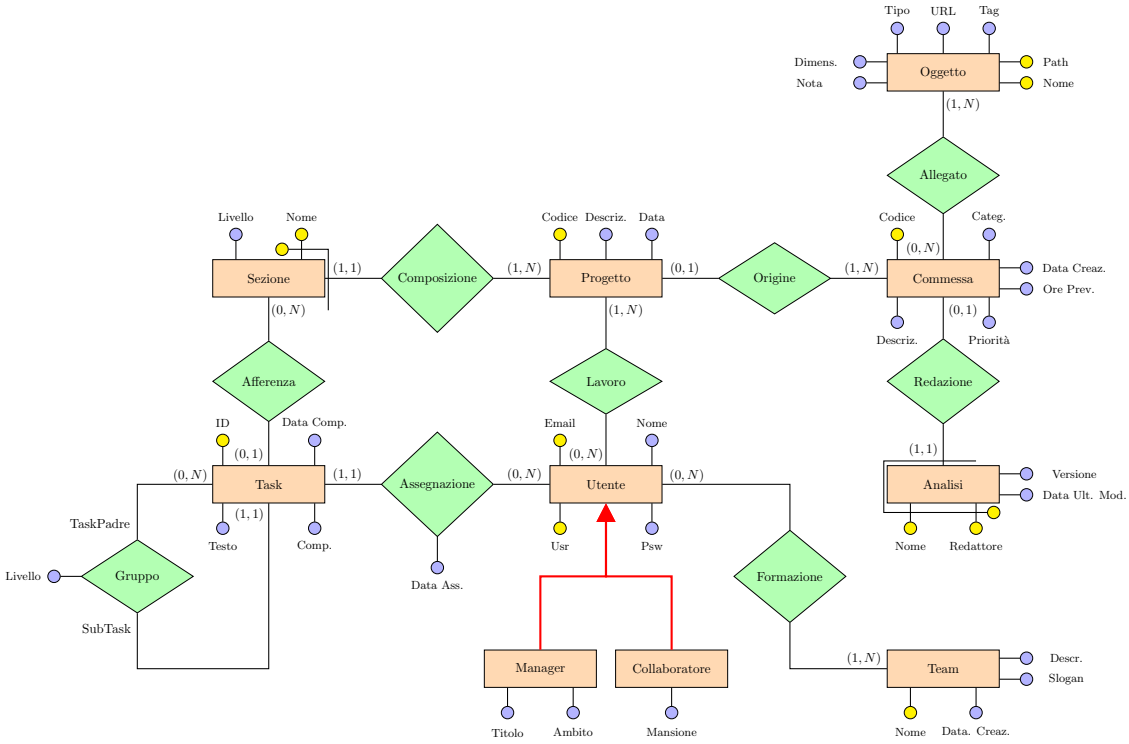
Anno Accademico 2023/2024

Indice

1 Introduzione

Il progetto di seguito descritto prevede la modellazione e realizzazione di un database atto alla persistenza strutturata dei dati di un Task Manager, ossia di un applicativo che consente l'organizzazione di attività (denominate `Task`), raggruppate in `Sezioni` e afferenti ad un `Progetto`. È stato previsto che la base di dati progettata possa essere tanto utilizzata per scopi amatoriali quanto per finalità professionali, per cui sono state contemplate sia la strutturazione di `Team` di lavoro che la gestione di un meccanismo di derivazione di un `Progetto` da una `Commessa`, la quale potrà essere corredata da un'`Analisi` e/o da uno o più `Oggetti` atti alla chiarificazione degli obiettivi progettuali.

2 Modello concettuale



2.1 Cardinalità

Di seguito si espongono i dettagli esplicativi in merito alle cardinalità delle relazioni di cui sopra:

- `Gruppo` - Una `Task` può essere a se stante oppure padre di un gruppo di più `SubTask` $(0, N)$, mentre una `SubTask`, in quanto tale, deve far riferimento ad una ed una sola `TaskPadre` $(1, 1)$;
- `Afferenza` - Una `Task` può essere a se stante oppure afferire ad al più una `Sezione` $(0, 1)$, mentre una `Sezione` può avere nessuna o più `Task` $(0, N)$;
- `Assegnazione` - Ad un `Utente` possono essere assegnate nessuna o più `Task` $(0, N)$, mentre una `Task` deve essere associata ad uno e un solo `Utente` $(1, 1)$;
- `Composizione` - Una `Sezione` deve far parte di un solo `Progetto` $(1, 1)$, mentre un `Progetto` deve contenere almeno una `Sezione` o più d'una $(1, N)$;
- `Lavoro` - Un `Utente` può lavorare a nessuno o a più `Progetti` $(0, N)$, mentre un `Progetto` deve essere associato ad almeno un `Utente`, ma anche a più d'uno $(1, N)$;

- `Formazione` - Un `Team` deve essere formato da almeno un `Utente` o da più d'uno $(1, N)$, mentre un `Utente` può non essere incluso in alcun `Team` o concorrere alla formazione di più di un `Team` $(0, N)$;
- `Origine` - Un `Progetto` può essere a se stante, oppure essere originato da al più una `Commessa` $(0, 1)$, mentre una `Commessa`, se presente, dà origine ad almeno un `Progetto` o a più d'uno $(1, N)$;
- `Redazione` - Ad una `Commessa` può essere eventualmente associata la redazione di un'`Analisi` $(0, 1)$, mentre un'`Analisi`, se predisposta, è collegata ad una e una sola `Commessa` $(1, 1)$;
- `Allegato` - Un `Oggetto` deve essere necessariamente allegato ad almeno una `Commessa`, ma può anche essere associato a più di una `Commessa` $(1, N)$; invece una `Commessa` può essere priva di `Allegati` o averne molteplici $(0, N)$.

2.2 Dizionario dei dati

Di seguito si forniscono dei documenti aggiuntivi atti a chiarire in maniera quanto più esplicita le entità e relazioni coinvolte all'interno della base di dati.

2.2.1 Entità

Si espone nel seguito il dizionario delle entità:

Entità	**Descrizione**	**Attributi**	**Identificatore**
`Task`	Compito/Attività da completare da parte di un `Utente`.	ID, Data Comp. (Data Completamento), Testo, Completata (Compl.)	ID
`Sezione`	`Sezione` di afferenza di una determinata `Task`.	Nome, Livello	Nome
`Progetto`	`Progetto` a cui lavorano uno o più `Utenti` e contenente `Sezioni` di `Task` da completare; un `Progetto` può essere originato da una `Commessa`.	Codice, Descrizione (Descriz.), Data	Codice
`Commessa`	Richiesta esterna da parte di un committente cui viene associato un `Progetto` (o più d'uno) atto alla sua gestione e completamento.	Codice, Categoria (Categ.), Descrizione (Descriz.), Priorità, Data Creazione (Data Creaz.), Ore Preventivate (Ore Prev.)	Codice
`Oggetto`	Allegato di una `Commessa` e risorsa atta all'integrazione della richiesta del committente.	Nome, Path, Tipo, URL, Tag, Nota, Dimensione (Dimens.)	Codice, Path
`Analisi`	Documento creato per analizzare la richiesta del committente e formulare le soluzioni progettuali.	Nome, Redattore, Versione, Data Ultima Modifica (Data Ult. Mod.)	Nome, Redattore
`Utente`	Attore attivo che può lavorare ad un `Progetto`, completare `Task` ed essere parte integrante di un `Team`.	Email, Username (Usr.), Nome, Password (Psw.)	Email, Username
`Manager`	Particolare `Utente` con maggiori responsabilità, avente un *Titolo* che ne certifica la competenza e un *Ambito* di specializzazione.	Email, Username (Usr.), Nome, Password (Psw.), Titolo, Ambito	Email, Username

`Collaboratore`	Particolare `Utente` che, nell'ambito di un `Team` o `Progetto`, svolge una propria *Mansione*.	Email, Username (Usr.), Nome, Password (Psw.), Mansione	Email, Username
`Team`	Gruppo di `Utenti` che cooperano per il raggiungimento di un obiettivo quale la realizzazione di un `Progetto`.	Nome, Descrizione (Descriz.), Slogan, Data Creazione (Data Creaz.)	Nome

2.2.2 Relazioni

Si espone nel seguito il dizionario delle relazioni:

Relazioni	**Descrizione**	**Componenti**	**Attributi**
`Gruppo`	Raggruppamento di `SubTask` in una `Task`.	`Task`	Livello
`Afferenza`	Afferenza di una `Task` ad una `Sezione`.	`Task`, `Sezione`	
`Composizione`	Composizione di un `Progetto` in `Sezioni`.	`Sezione`, `Progetto`	
`Assegnazione`	Assegnazione di una `Task` ad un dato `Utente`.	`Task`, `Utente`	Data Assegnazione (Data Ass.)
`Lavoro`	Lavoro ad un `Progetto` da parte di uno o più `Utenti`.	`Progetto`, `Utente`	
`Origine`	Origine di un `Progetto` da una `Commessa`.	`Progetto`, `Commessa`	
`Allegato`	Associazione di un `Oggetto` ad una `Commessa`.	`Oggetto`, `Commessa`	
`Redazione`	Redazione di un' `Analisi` associata ad una `Commessa`.	`Analisi`, `Commessa`	
`Formazione`	Formazione di un `Team` da parte degli `Utenti`.	`Team`, `Utente`	

2.3 Vincoli non esprimibili

I vincoli non esprimibili dal modello concettuale vengono di seguito esplicitati:

Vincolo	**Descrizione**
(1)	Un `Team` non deve essere composto da più di 3 `Manager`.
(2)	Un `Team` deve essere composto da `Manager` di ambiti differenti.
(3)	Ad un `Progetto` può lavorare al più 1 `Manager`.
(4)	La *Data Completamento* di una `Task` non deve essere antecedente alla *Data* della `Task` stessa.
(5)	Non possono coesistere due o più `SubTask` di una medesima `Task` che abbiano lo stesso *Livello*.
(6)	Non possono coesistere due o più `Sezioni` di un medesimo `Progetto` che abbiano lo stesso *Livello*.

2.4 Tavola dei volumi

Di seguito si espone la tavola dei volumi in riferimento al modello concettuale di cui sopra:

Concetto	**Tipo**	**Volume**
`Sezione`	E	50000
`Progetto`	E	10000
`Utente`	E	400
`Task`	E	2000000
`Oggetto`	E	850
`Commessa`	E	4500
`Analisi`	E	4000
`Team`	E	75
`Assegnazione`	R	2000000
`Afferenza`	R	1500000
`Composizione`	R	50000
`Lavoro`	R	17000
`Gruppo`	R	150000
`Origine`	R	5000
`Formazione`	R	230
`Allegato`	R	1200
`Redazione`	R	4000

2.5 Valutazione del costo

Per le operazioni di interesse, successivamente dettagliate, viene di seguito esposta la relativa valutazione del costo:

Operazione	**Tipo**	**Frequenza**
Selezionare le `Task` completate all'interno di uno specifico intervallo di date.	Interattiva	20/giorno
Inserire una nuova `Task` di una data `Sezione`; se taluna non è presente crearla e associarla al `Progetto` specificato.	Interattiva	10/giorno
Incrementare la *Versione* di tutte le `Analisi` che hanno una *Data Ultima Modifica* maggiore di una data specifica.	Batch	2/settimana
Estrapolare una statistica relativa alle `Task` completate/-da fare di ogni `Utente`.	Batch	10/settimana
Visualizzare, per ogni `Commessa` di una data *Categoria*, il numero di `Progetti` originati e la dimensione totale degli `Oggetti`	Interattiva	4/settimana
Selezionare, per tutti i `Progetti` di un dato `Utente`, le relative `Sezioni`, `Task`, `Commesse` e `Allegati`	Interattiva	1500/giorno

2.6 Ristrutturazione del modello concettuale

2.6.1 Analisi delle ridondanze

All'interno del modello concettuale di cui sopra non sono presenti degli attributi derivabili, ma essendovi un ciclo che coinvolge le entità `Task`, `Sezione`, `Progetto` e `Utente`, si palesa una ridondanza associata alla composizione di tali relazioni. Tuttavia

- La relazione `Assegnazione` non può essere eliminata, in quanto altrimenti non si potrebbero determinare le `Task` assegnate e/o completate da un dato `Utente`; un `Progetto`, infatti, deve essere composto da almeno una `Sezione`, ma ad uno stesso `Progetto` possono lavorare anche più `Utenti`; inoltre, una `Task` può anche non essere associata ad alcuna `Sezione`; ne deriva che tanto per `Task` prive di `Sezione` quanto per `Task` afferenti ad una specifica `Sezione` non sarebbe sempre possibile attribuire ad un `Utente` le proprie `Task` assegnate/completate.
- La relazione `Afferenza` non può essere eliminata, in quanto altrimenti non si potrebbero andare a determinare le `Task` afferenti ad una data `Sezione`: dall'estrapolazione delle `Sezioni` di un dato `Progetto` a cui lavora un `Utente` non è possibile definire con precisione inequivocabile quali `Task` a lui assegnate afferiscono a quale `Sezione`.
- La relazione `Composizione` non può essere eliminata, in quanto altrimenti non si potrebbero andare a determinare le `Sezioni` componenti di un dato `Progetto`: dall'estrapolazione delle `Sezioni` a cui afferiscono le `Task` assegnate ad un dato `Utente` non è possibile definire senza ambiguità quali `Sezioni` appartengono a quale `Progetto`.

L'unica relazione del ciclo che potrebbe essere eliminata è `Lavoro`, in quanto ricavando i `Progetti` contenenti le `Sezioni` a cui appartengono le `Task` assegnate ad un dato `Utente` si potrebbero determinare i `Progetti` a cui l'`Utente` lavora; si consideri, a tal proposito, la sotto-tavola dei volumi seguente

Concetto	**Tipo**	**Volume**
`Sezione`	E	50000
`Progetto`	E	10000
`Utente`	E	400
`Task`	E	2000000
`Assegnazione`	R	2000000
`Afferenza`	R	1500000
`Composizione`	R	50000
`Lavoro`	R	17000

e le pseudo-operazioni seguenti (che prevedono azioni parziali incluse nelle macro-operazioni succitate):

1. Associare un nuovo `Utente` ad un dato `Progetto` e assegnargli una nuova `Task` - 10 volte al giorno;
2. Visualizzare i `Progetti` a cui lavora un dato `Utente` - 1500 volte al giorno.

Allora, in presenza di ridondanza

1. Per la prima operazione

Concetto	**Costrutto**	**Accessi**	**Tipo**
`Utente`	Entità	1	S
`Lavoro`	Relazione	1	S
`Task`	Entità	1	S
`Assegnamento`	Relazione	1	S

2. Per la seconda operazione

Concetto	Costrutto	Accessi	Tipo
`Utente`	Entità	1	L
`Lavoro`	Relazione	25	L
`Progetto`	Entità	25	L

avendo calcolato che, in media, ogni `Utente` lavora a $\frac{10000}{400} = 25$ `Progetti`.

Contando doppi gli accessi in scrittura, la prima operazione comporta $(4 \times 10) \times 2 = 80$ accessi al giorno, mentre la seconda $(1 + 25 + 25) \times 1500 = 76500$ accessi al giorno.

Invece, in assenza di ridondanza

1. Per la prima operazione

Concetto	Costrutto	Accessi	Tipo
`Utente`	Entità	1	S
`Assegnamento`	Relazione	1	S
`Task`	Entità	1	S

2. Per la seconda operazione

Concetto	Costrutto	Accessi	Tipo
`Utente`	Entità	1	L
`Assegnazione`	Relazione	5000	L
`Task`	Entità	5000	L
`Afferenza`	Relazione	3750	L
`Sezione`	Entità	125	L
`Composizione`	Relazione	125	L
`Progetto`	Entità	25	L

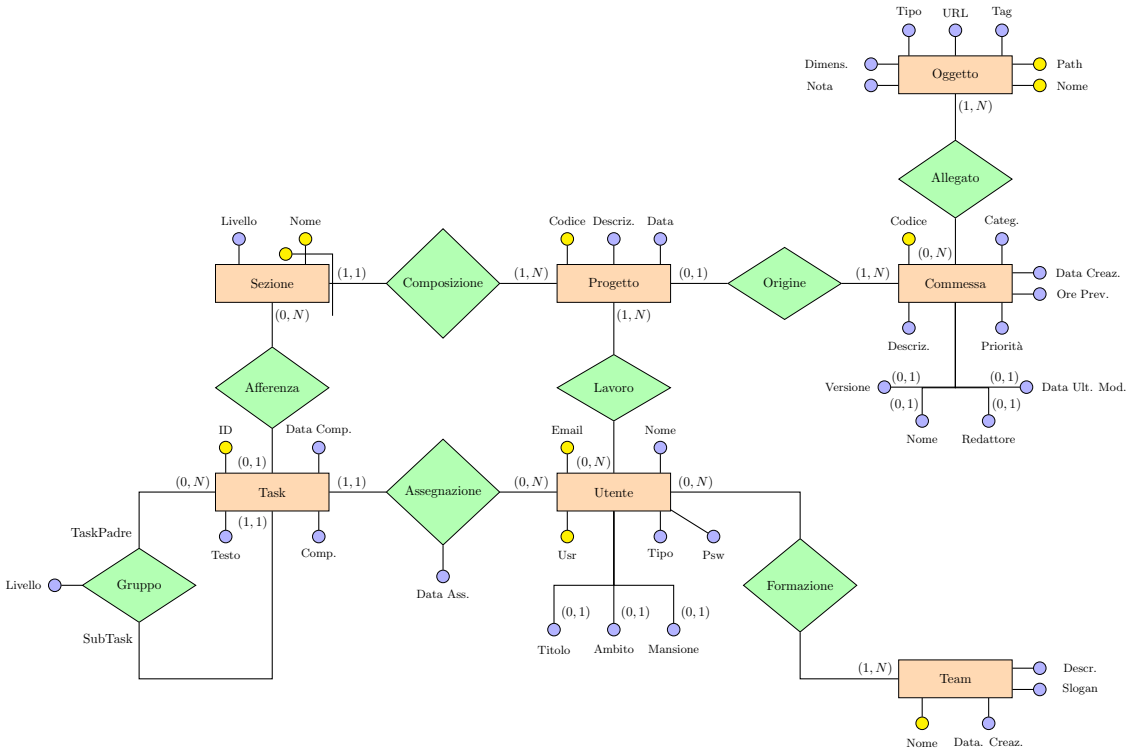
avendo calcolato che, in media, ad ogni `Utente` sono assegnate $\frac{2000000}{400} = 5000$ `Task`; inoltre, essendo 2000000 le `Task` e 1500000 le `Afferenze`, tre `Task` su quattro, ovvero 1500000 `Task` totali e 3750 per `Utente`, afferiscono ad una `Sezione`. Essendo 50000 le `Sezioni` e 1500000 le `Task` con `Sezione`, in media una `Sezione` contiene 30 `Task`, per cui 125 `Sezioni` sono indirettamente collegate ad un `Utente`; da ultimo, essendo 10000 i `Progetti` e 50000 le `Sezioni`, in media un `Progetto` contiene 5 `Sezioni`, per cui un `Utente` lavora in media a 25 `Progetti`.

Contando doppi gli accessi in scrittura, la prima operazione comporta $(3 \times 10) \times 2 = 60$ accessi al giorno, mentre la seconda $(1 + 5000 + 5000 + 3750 + 125 + 125 + 25) \times 1500 = 21039000$ accessi al giorno. Appare evidente, quindi, che convenga preservare la ridondanza delle relazioni.

2.6.2 Eliminazione delle generalizzazioni

Per la ristrutturazione del modello concettuale di cui sopra deve essere eliminata la generalizzazione tra `Utente` e i figli `Manager` e `Collaboratore`; per far ciò si adotta l'accorpamento delle entità figlie nell'entità padre, in quanto le entità figlie introducono differenziazioni non sostanziali e le operazioni d'accesso non distinguono tra occorrenze dell'entità padre e dell'entità figlia (rendendo l'accesso più efficiente): data una `Task`, un `Progetto` o un `Team`, la conoscenza della particolarità dell'`Utente` partecipante nella relazione concorre alla contestualità dell'accesso risultante.
Oltre a ciò, date le entità `Commessa` e `Analisi`, legate dalla relazione uno a uno `Redazione`, siccome

le operazioni che si andranno eseguendo coinvolgeranno sempre tali entità contestualmente, senza mai avere accessi separati, si ritiene conveniente accorpare le due entità nell'entità `Commessa`.
Il modello concettuale ristrutturato è



3 Modello logico

Il modello logico risultante dalla traduzione del modello concettuale ristrutturato è

`Task`(ID, Testo, DataCompletamento, Completata, Livello, TaskPadre, Sezione, Email, Username, DataAssegnazione)

`Sezione`(Nome, Progetto, Livello)

`Progetto`(Codice, Descrizione, Data, Commessa)

`Utente`(Email, Username, Nome, Password, Tipo, Titolo, Ambito, Mansione)

`Lavoro`(Progetto, UtenteEmail, UtenteUsername)

`Commessa`(Codice, Categoria, Descrizione, Priorità, DataCreazione, OrePreventivate, Versione, Nome, Redattore, DataUltimaModifica)

`Oggetto`(Path, Nome, Tipo, URL, Tag, Dimensione, Nota)

`Allegato`(Commessa, OggettoPath, OggettoNome)

`Team`(Nome, DataCreazione, Descrizione, Slogan)

`Formazione`(Team, UtenteEmail, UtenteUsername)

4 Normalizzazione

Dal modello logico di cui sopra si evince che

- la **prima forma normale** (1NF) è rispettata, in quanto ogni relazione (tabella) ha colonne atomiche, per cui si ha una colonna per valore senza unità ripetitive.
- la **seconda forma normale** (2NF) è rispettata, in quanto ogni relazione (tabella) soddisfa i requisiti della 1NF e tutti gli attributi non-chiave presentano una dipendenza funzionale dalla chiave nella sua interezza; per le tabelle con chiavi composte, infatti, ogni attributo non dipende solo da una parte della chiave, ma dall'intera chiave.
- la **terza forma normale** (3NF) è rispettata, in quanto ogni relazione (tabella) soddisfa i requisiti della 2NF e contiene solo colonne non dipendenti in modo transitivo dalla chiave primaria.

Osservazione: Per evitare scorrettezze in fase di inserimento di dati non numerici, si potrebbero, eventualmente, andare a creare delle tabelle di lookup che associno ad ogni informazione testuale un corrispettivo identificativo numerico. Per esempio, nel caso della tabella `Commessa`, tanto *Priorità* quanto *Redattore* sono delle colonne testuali per le quali si desidererebbe standardizzarne il formato, scongiurando inserimenti disomogenei:

ID	Priorità
1	Bassa
2	Media
3	Alta

ID	Redattore
1	Mario Rossi
2	Luigi Bianchi
3	Paolo Gialli

5 Operazioni

Alla luce di quanto sopra esposto, saranno di interesse le seguenti operazioni:

1. Selezionare le `Task` completate all'interno di uno specifico intervallo di date

```
DELIMITER $$
CREATE PROCEDURE sp_getCompletedTask(IN daData DATE, IN aData DATE)
BEGIN
    SELECT Testo, DataAssegnazione, CONCAT('Completata da ',
    IFNULL(Username, 'Nessuno'),
    ' - ',
    IFNULL(Email, 'Nessuno'),
    ' il giorno ',
    DATE_FORMAT(DataCompletamento, '%d/%m/%Y')) AS Completamento
    FROM Task
    WHERE Completata = TRUE AND DataCompletamento BETWEEN daData AND
        aData;
END $$
DELIMITER ;

CALL sp_getCompletedTask('20240101', '20241231');
```

2. Inserire una nuova `Task` di una data `Sezione`; se taluna non è presente crearla e associarla al `Progetto` specificato

```
1   DELIMITER $$
2   CREATE PROCEDURE sp_insertTask(IN parTesto TEXT, IN parTaskPadre INT, IN
        parSezione VARCHAR(50), IN parProgetto INT, IN parEmail
        VARCHAR(100), IN parUsername VARCHAR(100))
3   BEGIN
4       DECLARE EXIT HANDLER FOR SQLEXCEPTION
5       BEGIN
6           ROLLBACK;
7           SELECT 'Errore: Impossibile completare l''inserimento della Task';
8       END;
9
10      START TRANSACTION;
11          SELECT @varProgetto := IFNULL(MAX(P.Codice),-1) FROM Progetto AS P
12          LEFT JOIN Sezione AS S ON S.Progetto = P.Codice
13          WHERE S.Nome = parSezione;
14
15          IF @varProgetto = -1 THEN
16          SELECT @varSezMaxLiv := IFNULL(MAX(S.Livello),-1) FROM Progetto
                AS P
17          LEFT JOIN Sezione AS S ON S.Progetto = P.Codice
18          WHERE P.Codice = parProgetto;
19
20          IF @varSezMaxLiv = -1 THEN
21              SIGNAL SQLSTATE '45000'
22              SET MESSAGE_TEXT = 'Nessuna Sezione trovata per il Progetto.
                    Progetto inesistente!';
23          ELSE
24              -- Nuova sezione
25              INSERT INTO Sezione VALUES (parSezione, parProgetto,
                    @varSezMaxLiv+1);
26          END IF;
27          END IF;
28
29          SELECT @varTaskMaxLiv := IFNULL(MAX(T.Livello),0)+1 FROM Task AS T
30          LEFT JOIN Sezione AS S ON S.Nome = T.Sezione
31          WHERE T.Sezione = parSezione;
32
33          -- Nuova task
34          INSERT INTO Task VALUES (NULL, parTesto, NOW(), FALSE,
                @varTaskMaxLiv, parTaskPadre, parSezione, parEmail,
                parUsername, NULL);
35
36      COMMIT;
37  END $$
38  DELIMITER ;
39
40  CALL sp_insertTask("Testo per nuova task", 1, "Nuova Sezione", 1,
        "user1@db.it", "user1");
```

3. Estrapolare una statistica relativa alle `Task` completate/da fare di ogni `Utente`

```sql
CREATE VIEW V_StatTask AS
    SELECT U_EX.Email, U_EX.Username, SumCompletate AS Completate,
        SumDaFare AS DaFare,
    TotTask AS TaskTotali, SumCompletate / TotTask AS
        PercentualeCompletate,
    SumCompletate / DiffDate AS PercentualeCompletateGiornaliere
    FROM Utente AS U_EX
    INNER JOIN (SELECT U_IN.Email, U_IN.Username, SUM(CASE WHEN
        Completata = TRUE THEN 1 ELSE 0 END) AS SumCompletate,
        SUM(CASE WHEN Completata = FALSE THEN 1 ELSE 0 END) AS SumDaFare,
        COUNT(*) AS TotTask, MAX(T.DataAssegnazione) -
            MIN(T.DataAssegnazione) AS DiffDate
        FROM Utente AS U_IN
        INNER JOIN Task AS T ON T.Email = U_IN.Email AND T.Username =
            U_IN.Username
        GROUP BY U_IN.Email, U_IN.Username) AS TabSumTask ON
            TabSumTask.Email = U_EX.Email AND TabSumTask.Username =
            U_EX.Username;

SELECT * FROM V_StatTask;
```

4. Controllare che, all'inserimento di un `Team`, esso abbia al più 3 `Manager`, i quali devono essere di *Ambiti* diversi

```sql
DELIMITER $$
CREATE TRIGGER trg_TeamManager
BEFORE INSERT ON Formazione
FOR EACH ROW
BEGIN
    DECLARE numManager INT;
    DECLARE numManagerAmbiti INT;

    SET numManager = (SELECT COUNT(U.Tipo) FROM Formazione AS F
    INNER JOIN Utente AS U USING(Email, Formazione)
    WHERE F.Team = NEW.Team AND U.Tipo = 'Manager');

    SET numManagerAmbiti = (SELECT COUNT(U.Ambito) FROM Formazione AS F
    INNER JOIN Utente AS U USING(Email, Formazione)
    WHERE F.Team = NEW.Team AND U.Tipo = 'Manager'
    ORDER BY COUNT(Ambito) DESC
    LIMIT 1);

    IF numManager > 3 THEN
        SIGNAL SQLSTATE '45001'
        SET MESSAGE_TEXT = 'Il Team ha gia'' 3 Manager';
    ELSE IF numManagerAmbiti > 1 THEN
        SIGNAL SQLSTATE '45002'
            SET MESSAGE_TEXT = 'Il Team ha gia'' un Manager per l''Ambito
                previsto';
        END IF;
    END IF;
END $$
DELIMITER ;
```

5. Visualizzare, per ogni `Commessa` di una data *Categoria*, i dettagli della relativa *Analisi*, il numero di `Progetti` originati e la *Dimensione* totale degli `Oggetti`:

```
PREPARE st_CommData FROM
    'SELECT C.Codice, C.Categoria, C.Nome AS Analisi, C.Versione,
        COUNT(*) AS NumProgetti, SUM(O.Dimensione)
     FROM Commessa AS C LEFT JOIN Progetto AS P ON C.Codice = P.Commessa
     LEFT JOIN Allegato AS A ON C.Codice = A.Commessa
     LEFT JOIN Oggetto AS O ON O.Path = A.OggettoPath AND O.Nome =
         A.OggettoNome
     WHERE C.Categoria = ?
     GROUP BY C.Codice, C.Categoria, C.Nome, C.Versione';

SET @varCat = 'Categoria_1';
EXECUTE st_CommData USING @varCat;

DEALLOCATE PREPARE st_CommData;
```

6. Selezionare, per tutti i `Progetti` di un dato `Utente`, le relative `Sezioni`, `Task`, `Commesse` e `Allegati`

```
CREATE VIEW V_Progetti AS
    SELECT U.Email, U.Username, P.Codice AS CodProg, P.Descrizione AS
        DescProg, P.Data AS DataProg, S.Nome AS Sezione, T.Testo AS
        Task, T.DataAssegnazione AS DataAssTask, T.Completata AS
        Completata, T.DataCompletamento AS DataCompTask, C.Codice AS
        CodCom, C.Categoria AS CatCom, C.Nome AS Analisi, C.Versione,
        C.Redattore, O.Path AS PathOggetto, O.Nome AS NomeOggetto,
        O.Dimensione AS DimOggetto
    FROM Utente U
    INNER JOIN Lavoro L ON U.Email = L.Email AND U.Username = L.Username
    INNER JOIN Progetto P ON L.Progetto = P.Codice
    INNER JOIN Sezione S ON P.Codice = S.Progetto
    INNER JOIN Task T ON S.Nome = T.Sezione
    INNER JOIN Commessa C ON P.Codice = C.Codice
    INNER JOIN Allegato A ON A.Commessa = C.Codice
    INNER JOIN Oggetto O ON O.Path = A.OggettoPath AND O.Nome =
        A.OggettoNome

  PREPARE st_getProgByUser FROM 'SELECT * FROM V_Progetti WHERE Email =
      ? AND Username = ?';

  SET @mail = 'user1@db.it';
  SET @user = 'user1';
  EXECUTE st_getProgByUser USING @mail, @user;

  DEALLOCATE PREPARE st_getProgByUser;
```

7. Incrementare la *Versione* di tutte le `Analisi` che hanno una *Data Ultima Modifica* maggiore di una data specifica:

```
PREPARE st_updateVersion FROM 'UPDATE Commessa SET Versione = Versione +
    1 WHERE DataUltimaModifica > ?';

SET @varData = '2024-01-01';
EXECUTE st_updadeVersion USING @varData;

DEALLOCATE PREPARE st_updateVersion;
```

8. Creare una funzione che, date le *Ore Preventivate* di una `Commessa`, la sua *Data Creazione* e la *Data Ultima Modifica* dell'`Analisi` associata, restituisca la *Priorità* della commessa stessa:

```
1   DELIMITER $$
2   CREATE FUNCTION udfn_getPriority(parOrePreventive INT, parDataCreazione
        DATE, parDataUltimaMod DATE)
3   RETURNS INT DETERMINISTIC
4   BEGIN
5       DECLARE varPriorita INT;
6
7       IF parOrePreventive > 10 THEN
8       SET varPriorita = 3;
9       ELSEIF parOrePreventive > 5 AND ABS(DATEDIFF(parDataUltimaMod,
            parDataCreazione)) < 20 THEN
10      SET varPriorita = 3;
11      ELSEIF parOrePreventive > 5 AND ABS(DATEDIFF(parDataUltimaMod,
            parDataCreazione)) > 20 THEN
12      SET varPriorita = 2;
13      ELSE
14      SET varPriorita = 1;
15      END IF;
16
17      RETURN varPriorita;
18  END $$
19  DELIMITER ;
20
21  SELECT udfn_getPriority(5, '2024-01-01', '2024-01-15');
```