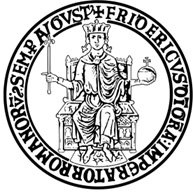
Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell’Informazione



Corso di Laurea in Informatica

Insegnamento di Basi di Dati I

Anno Accademico 2020/2021

Progettazione e sviluppo di una base di dati relazionale per un sistema di planning per la gestione di progetti.

Autori:

Chehade Bianca Giada

Matricola N86003209

[b.chehade@studenti.unina.it](mailto:b.chehade@studenti.unina.it)

Zaza Francesco Rosario

Matricola N86002501

fra.zaza@studenti.unina.it

*Questa pagina è stata lasciata intenzionalmente bianca.*

Indice

[1. Descrizione del progetto 4](#_Toc66037448)

[1.1 Descrizione sintetica 5](#_Toc66037449)

[2. Progettazione concettuale 5](#_Toc66037450)

[2.1 Class diagram 6](#_Toc66037451)

[2.2 Ristrutturazione del class diagram 6](#_Toc66037452)

[2.2.1 Analisi delle ridondanze 6](#_Toc66037453)

[2.2.2 Analisi degli identificativi 6](#_Toc66037454)

[2.2.3 Rimozione degli attributi multipli 6](#_Toc66037455)

[2.2.4 Rimozione delle gerarchie di specializzazione 7](#_Toc66037456)

[2.3 Class diagram ristrutturato 7](#_Toc66037457)

[2.4 Dizionario dei dati 7](#_Toc66037458)

[2.4.1 Dizionario delle classi 7](#_Toc66037459)

[2.4.2 Dizionario delle associazioni 9](#_Toc66037460)

[2.4.3 Dizionario dei vincoli 11](#_Toc66037461)

[3. Progettazione logica 11](#_Toc66037462)

[3.1 Traduzione in schemi relazionali 12](#_Toc66037463)

[3.1.1 Traduzione delle associazioni 12](#_Toc66037464)

[3.2 Schema Logico 13](#_Toc66037465)

[4. Progettazione fisica 14](#_Toc66037466)

[4.1 Definizione delle tabelle 14](#_Toc66037467)

[4.1.1 Definizione della tabella **Azienda** 14](#_Toc66037468)

[4.1.2 Definizione della tabella **Privato** 14](#_Toc66037469)

[4.1.3 Definizione della tabella **Società** 14](#_Toc66037470)

[4.1.4 Definizione della tabella **Progetto** 14](#_Toc66037471)

[4.1.5 Definizione della tabella **Partecipante** 15](#_Toc66037472)

[4.1.6 Definizione della tabella **ProgRealizzato** 15](#_Toc66037473)

[4.1.7 Definizione della tabella **PartecipanteProg** 15](#_Toc66037474)

[4.1.8 Definizione della tabella **Meeting** 15](#_Toc66037475)

[4.1.9 Definizione della tabella **CompMeeting** 15](#_Toc66037476)

[4.1.10 Definizione della tabella **Ambito** 16](#_Toc66037477)

[4.1.11 Definizione della tabella **ProgAmbito** 16](#_Toc66037478)

Capitolo 1

# 1. Descrizione del progetto

## 1.1 Descrizione sintetica

Si svilupperà ed implementerà una base di dati relazionale utile alla gestione di progetti in un’azienda. Tale database prevede la possibilità di tenere traccia dei partecipanti a ciascun progetto, identificando il ruolo per ognuno di essi (per ogni progetto ci sarà un solo project manager). Ad ogni progetto è associato una tipologia (“Ricerca di base”, “Ricerca Industriale”, “Ricerca sperimentale”, “Sviluppo Sperimentale”, ...) ed uno o più ambiti (Economia, Medicina, …). Il sistema permetterà anche l'organizzazione di meeting fisicamente, in sale riunioni, o telematicamente su una piattaforma di videoconferenza. Si terrà traccia delle partecipazioni ai progetti ed ai meeting, ai fini della valutazione del singolo partecipante. In fase di creazione di un nuovo progetto, i partecipanti dovranno essere selezionati in base a criteri di ricerca che includono anche il salario medio e la valutazione aziendale del partecipante, oltre alla tipologia di progetti cui ha preso parte.

Capitolo 2

# 2. Progettazione concettuale

In questo capitolo inizia la progettazione della base di dati a livello concettuale. Dal risultato dell’analisi dei requisiti che devono essere soddisfatti si arriverà ad uno schema concettuale indipendente dalla struttura dei dati e dall’implementazione fisica. Tale schema concettuale sarà rappresentato usando un class diagram UML, nel quale saranno evidenziate le entità rilevanti ai fini della rappresentazione dei dati e le relazioni che intercorrono tra esse.

## 2.1 Class diagram

//inserisci class diagram

## 2.2 Ristrutturazione del class diagram

Si procede alla ristrutturazione del class diagram con lo scopo di renderlo idoneo alla traduzione in schemi relazionali e di migliorare l’efficienza dell’implementazione. Al termine del procedimento il class diagram non conterrà attributi strutturati, attributi multipli e gerarchie di specializzazione.

### 2.2.1 Analisi delle ridondanze

Non sono presenti ridondanze significative da eliminare.

### 2.2.2 Analisi degli identificativi

Si procede all’aggiunta, per alcune entità, di chiavi “surrogate”. Tali attributi saranno identificativi numerici che permetteranno più agevolmente un’identificazione univoca per ciascuna istanza.

### 2.2.3 Rimozione degli attributi multipli

L’attributo **Ambito** della classe **Progetto** è da eliminare. Per andare incontro alla necessità di avere più ambiti possibili per un determinato progetto, è ragionevole procedere alla creazione di una nuova classe **Ambito;** la cardinalità molti a molti della nuova associazione tra **Progetto** e **Ambito** sarà risolta con l’implementazione della classe **ProgAmbito**

Lo stesso procedimento è da eseguire per l’attributo multiplo **ProgettiRealizzati** della classe **Partecipante;** si procede alla creazione delle classi **ProgRealizzato e PartecipanteProg** (utile data la cardinalità molti a molti dell’associazione tra **Partecipante** e **ProgRealizzato**).

### 2.2.4 Rimozione delle gerarchie di specializzazione

Si procede con l’eliminazione delle specializzazioni della classe **Cliente**. Si tratta di una specializzazione totale e disgiunta, dunque si procederà all’eliminazione “schiacciando” la superclasse nelle sottoclassi.

## 2.3 Class diagram ristrutturato

//INSERISCI CLASS DIAGRAM RISTRUTTURATO

## 2.4 Dizionario dei dati

### 2.4.1 Dizionario delle classi

|  |  |  |
| --- | --- | --- |
| Classe | Descrizione | Attributi |
| Azienda | Descrittore dell’azienda che realizza un progetto. | **ISIN** (varchar): chiave primaria. Codice identificativo univoco per le aziende.  **Nome** (varchar): nome dell'azienda.  **SedePrincipale** (varchar): indica dove è situata l'azienda. |
| Progetto | Descrittore del progetto realizzato da un’azienda e commissionato da un privato/società. | **CodProgetto** (integer): identifica univocamente ogni istanza di **Progetto**.  **Tipologia** (Enumerazione): indica la tipologia del progetto.  **NumeroPartecipanti** (integer): specifica quante persone lavorano al progetto.  **Budget** (decimal): specifica il costo totale del progetto. |
| Partecipante | Descrive ciascun partecipante al progetto. | **UserID** (integer): chiave surrogata. Identificativo numerico di ciascun partecipante al progetto.  **Email** (varchar, opzionale): specifica l’indirizzo di posta elettronica delpartecipante.  **Password** (varchar, opzionale): indica la sequenza di caratteri alfanumerici necessaria per accedere alla email.  **CF** (varchar): identifica in modo univoco le persone fisiche che partecipano al progetto.  **Nome** (varchar): nome del partecipante.  **Cognome** (varchar): cognome del partecipante.  **Ruolo** (enumerazione): specifica il ruolodel dipendente nel progetto a cui sta lavorando.  **SalarioMedio** (decimal): indica il guadagno medio di un partecipante.  **Valutazione** (integer, opzionale): fornisce una valutazione numerica di un partecipante. |

*Continua nella pagina seguente.*

|  |  |  |
| --- | --- | --- |
| Classe | Descrizione | Attributi |
| Società | Descrittore della società che commissiona il progetto. | **PartitaIVA** (varchar):sequenza di cifre che identifica univocamente un soggetto che esercita un’attività.  **NomeSocietà** (varchar): specifica il nome della società. |
| Privato | Descrittore del privato che commissiona il progetto. | **CF** (varchar): identifica in modo univoco le persone fisiche.  **Nome** (varchar): nome del privato.  **Cognome** (varchar): cognome del privato. |
| Meeting | Descrittore delle riunioni sia fisiche che telematiche. | **CodMeeting** (integer): sequenza numerica che permette di identificare univoamente la riunione.  **DataRiunione** (date): indica il giorno della riunione. **OraInizio** (time): indica l’ora di inizio della riunione. **OraFine** (time): indica l’ora di fine della riunione.  **Luogo** (varchar, opzionale): indica il luogo presso il quale si svolge la riunione.  **Piattaforma** (varchar, opzionale): indica la piattaforma utilizzata nel caso in cui il meeting si tenga in modalità telematica. |
| Ambito | Descrittore dell’ambitodi un progetto. | **Nome** (enumerazione): indica in che ambito è indirizzato il progetto in questione. |
| ProgRealizzato | Descrittore dei progetti realizzati in passato dai progettisti. | **CodProg** (integer): chiave surrogata; codice identificativo di un progetto.  **Tipologia** (enumerazione): indica di che tipo è il progetto. |

### 2.4.2 Dizionario delle associazioni

|  |  |  |
| --- | --- | --- |
| Nome | Descrizione | Classi Coinvolte |
| Dipendenza | Esprime l’appartenenza di un partecipante ad una azienda. | **Partecipante[1.. \*]** ruolo **Lavora presso**: indica in quale azienda lavora un partecipante.  **Azienda[1]** ruolo **Impiega**: indica l’azienda presso la quale il partecipante lavora. |
| Gestione | Esprime da quale azienda è gestito un progetto. | **Progetto [1.. \*]** ruolo **Realizzato da**: indica da quale azienda è realizzato un progetto. **Azienda[1]** ruolo **Conduce:** indica il progetto che realizza un’azienda. |
| Realizzazione | Esprime l’impiego di un partecipante in un progetto. | **Progetto[1]** ruolo **E’ realizzato da**: indica da quale partecipante è realizzato il progetto.  **Partecipanti[1..\*]** ruolo **Partecipa**: indica a quale progetto partecipa un progettista. |
| PrivatoCommissiona | Esprime la commissione di un progetto da parte di un privato. | **Privato[1]** ruolo **Commissiona**: indica quale progetto commissiona un privato. **Progetto[1.. \*]** ruolo **Commissionato da**: indica da chi è commissionato il progetto. |
| SocietàCommissiona | Esprime la commissione di un progetto da parte di una società. | **Società[1]** ruolo **Commissiona**: indica quale progetto commissiona una società.  **Progetto[1.. \*]** ruolo **Commissionato da**: indica da chi è commissionato il progetto. |
| Argomento | Esprime la relazione tra un progetto e il suo ambito. | **Progetto[1]** ruolo **Composto da**: indica l’ambito del progetto. **ProgAmbito[\*]** ruolo **Compone:** indica a quale progetto si riferisce un ambito. |
| Caratterizzazione | Esprime la relazione tra un progetto e il suo ambito. | **Ambito[1]** ruolo **comprende**: indica che progetto comprende ciascun ambito.  **ProgAmbito[\*]** ruolo **Compreso in**:indica in che ambito è compreso un progetto |

*Continua nella pagina seguente.*

|  |  |  |
| --- | --- | --- |
| Nome | Descrizione | Classi Coinvolte |
| Partecipazione | Esprime l’appartenenza di un partecipante a un progetto. | **Partecipante[1]** ruolo **ha trattato**: indica il tipo di progetti ai quali ha lavorato un progettista. **PartecipanteProg[\*]** ruolo **trattato da**: indica da quale partecipante è stato realizzato un progetto. |
| ProgettoPrecedente | Esprime la relazione tra un partecipante e i progetti ai quali ha lavorato in precedenza. | **ProgRealizzato[1]** ruolo **Appartiene a**: indica a quale partecipante appartiene un progetto. **PartecipanteProg[\*]** ruolo **Composto da**: indica da quale partecipante è stato realizzato un progetto. |
| ConferenceCall | Esprime la relazione tra una riunione e i suoi partecipanti. | **Partecipante[1]** ruolo **Fa parte di**: indica a quale riunione partecipa un progettista. **CompMeeting[\*]** ruolo **Ospita**: indica quali progettisti ospita un meeting. |
| Composizione | Descrive le caratteristiche di ciascun meeting, | **Meeting[1]** ruolo **Composto da**: indica a quale meeting si riferisce la composizione.  **CompMeeting[\*]** ruolo **Compone**: indica quale meeting ha le caratteristiche in questione. |

### 2.4.3 Dizionario dei vincoli

Capitolo 3

# 3. Progettazione logica

In questo capitolo sarà trattata la fase successiva della progettazione della base di dati: si tradurrà lo schema concettuale in uno schema logico. Negli schemi relazionali che seguiranno le chiavi primarie sono indicate con una singola sottolineatura mentre le chiavi esterne con una doppia sottolineatura.

## 3.1 Traduzione in schemi relazionali

|  |
| --- |
| **Azienda** (ISIN, Nome, SedePrincipale). |

Chiavi esterne: nessuna.

|  |
| --- |
| **Privato** (CF, Nome, Cognome) |

Chiavi esterne: nessuna.

|  |
| --- |
| **Società** (PartitaIVA, NomeSocietà) |

Chiavi esterne: nessuna.

|  |
| --- |
| **Progetto** (CodProgetto, Tipologia, NumeroPartecipanti, Budget, ISIN, CF, PartitaIVA). |

Chiavi esterne: ISIN → Azienda.ISIN; CF → Privato.CF; PartitaIVA → Società.PartitaIVA.

|  |
| --- |
| **Partecipante** (UserID, Email, Pw, CF, Nome, Cognome, Ruolo, SalarioMedio, Valutazione, CodProgetto, ISIN). |

Chiavi esterne: CodProgetto → Progetto.CodProgetto; ISIN → Azienda.ISIN.

|  |
| --- |
| **ProgRealizzato** (CodProg, Tipologia) |

Chiavi esterne: nessuna.

|  |
| --- |
| **PartecipanteProg** (CodProg, UserID). |

Chiavi esterne: CodProg → ProgRealizzato.CodProg; UserID → Partecipante.UserID.

|  |
| --- |
| **Meeting** (CodMeeting, DataRiunione, OraInizio, OraFine, Piattaforma, Luogo). |

Chiavi esterne: nessuna.

|  |
| --- |
| **CompMeeting** (CodMeeting, UserID). |

Chiavi esterne: CodMeeting → Meeting.CodMeeting; UserID → Partecipante.UserID.

|  |
| --- |
| **Ambito** (Nome). |

Chiavi esterne: nessuna

|  |
| --- |
| **ProgAmbito** (CodProgetto, Nome) |

Chiavi esterne: CodProgetto → Progetto.CodProgetto; Nome → Ambito.Nome.

### 3.1.1 Traduzione delle associazioni

|  |  |
| --- | --- |
| Associazione | Implementazione |
| Dipendenza | Chiave esterna in **Partecipante** −→ **Azienda** |
| Gestione | Chiave esterna in **Progetto** −→ **Azienda** |
| Realizzazione | Chiave esterna in **Partecipante** −→ **Progetto** |
| PrivatoCommissiona | Chiave esterna in **Progetto** −→ **Privato** |
| SocietàCommissiona | Chiave esterna in **Progetto** −→ **Società** |
| Argomento | Chiave esterna in **ProgAmbito** −→ **Progetto** |
| Caratterizzazione | Chiave esterna in **ProgAmbito** −→ **Ambito** |
| Partecipazione | Chiave esterna in **PartecipanteProg** −→ **Progetto** |
| ProgettoPrecedente | Chiave esterna in **PartecipanteProg** −→ **ProgRealizzato** |
| ConferenceCall | Chiave esterna in **CompMeeting** −→ **Partecipante** |
| Composizione | Chiave esterna in **CompMeeting** −→ **Meeting** |

## 3.2 Schema Logico

In base a quanto detto nella sezione precedente, si previene al seguente schema logico.

|  |  |
| --- | --- |
| **Azienda** | (ISIN, Nome, SedePrincipale) |
| **Privato** | (CF, Nome, Cognome) |
| **Società** | (PartitaIVA, NomeSocietà) |
| **Progetto** | (CodProgetto, Tipologia, NumeroPartecipanti, Budget, ISIN, CF, PartitaIVA) |
| **Partecipante** | (UserID, Email, Pw, CF, Nome, Cognome, Ruolo, SalarioMedio, Valutazione, CodProgetto, ISIN). |
| **ProgRealizzato** | (CodProg, Tipologia) |
| **PartecipanteProg** | (CodProg, UserID). |
| **Meeting** | (CodMeeting, DataRiunione, OraInizio, OraFine, Piattaforma, Luogo). |
| **CompMeeting** | (CodMeeting, UserID). |
| **Ambito** | (Nome). |
| **ProgAmbito** | (CodProgetto, Nome) |

Capitolo 4

# 4. Progettazione fisica

## 4.1 Definizione delle tabelle

### 4.1.1 Definizione della tabella **Azienda**

1. CREATE TABLE Azienda (
2. ISIN varchar(12) PRIMARY KEY,
3. Nome varchar(12) NOT NULL,
4. SedePrincipale varchar(12) NOT NULL
5. );

### 4.1.2 Definizione della tabella **Privato**

1. CREATE TABLE Privato (
2. CF varchar(16) PRIMARY KEY,
3. Nome varchar(12) NOT NULL,
4. Cognome varchar(20) NOT NULL,
5. CONSTRAINT controllo\_cf CHECK (CF ~\* '[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]')
6. );

### 4.1.3 Definizione della tabella **Società**

1. CREATE TABLE Società (
2. PartitaIVA varchar(11) PRIMARY KEY,
3. NomeSocietà varchar(25) NOT NULL
4. );

### 4.1.4 Definizione della tabella **Progetto**

1. CREATE TABLE Progetto(
2. CodProgetto integer PRIMARY KEY,
3. Tipologia enum\_tipologia NOT NULL,
4. NumeroPartecipanti integer NOT NULL CHECK (numeropartecipanti > 0 AND numeropartecipanti <= 30),
5. Budget decimal NOT NULL,
6. ISIN varchar(12) NOT NULL REFERENCES Azienda(ISIN),
7. CF varchar(16) REFERENCES Privato(CF),
8. PartitaIVA varchar(11) REFERENCES Società(PartitaIVA),
9. CONSTRAINT vincolo\_cliente CHECK ((CF IS NOT NULL AND PartitaIva IS NULL) OR (CF IS NULL AND PartitaIva IS NOT NULL))
10. );

### 4.1.5 Definizione della tabella **Partecipante**

1. CREATE TABLE Partecipante (
2. UserID integer PRIMARY KEY,
3. Email varchar(30),
4. Pw varchar(30),
5. CF varchar(16) NOT NULL,
6. Nome varchar(20) NOT NULL,
7. Cognome varchar(20) NOT NULL,
8. Ruolo enum\_ruolo NOT NULL,
9. SalarioMedio decimal NOT NULL,
10. Valutazione integer,
11. CodProgetto integer NOT NULL REFERENCES Progetto(CodProgetto),
12. ISIN varchar(12) NOT NULL REFERENCES Azienda(ISIN)
13. CONSTRAINT controllo\_cf CHECK (CF ~\* '[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]')
14. );

### 4.1.6 Definizione della tabella **ProgRealizzato**

1. CREATE TABLE ProgRealizzato (
2. CodProg integer PRIMARY KEY,
3. Tipologia enum\_tipologia NOT NULL
4. );

### 4.1.7 Definizione della tabella **PartecipanteProg**

1. CREATE TABLE PartecipanteProg (
2. Username integer NOT NULL REFERENCES Partecipante(Username),
3. CodProg integer NOT NULL REFERENCES ProgRealizzato(CodProg)
4. );

### 4.1.8 Definizione della tabella **Meeting**

1. CREATE TABLE Meeting(
2. CodMeeting integer PRIMARY KEY,
3. DataRiunione date NOT NULL,
4. OraInizio time NOT NULL,
5. OraFine time,
6. Piattaforma varchar(10),
7. Luogo varchar(10),
8. CONSTRAINT luogo\_riunione CHECK ( (Luogo IS NOT NULL AND Piattaforma IS NULL) OR (Luogo IS NULL AND Piattaforma IS NOT NULL)
9. );

### 4.1.9 Definizione della tabella **CompMeeting**

1. CREATE TABLE CompMeeting (
2. CodMeeting integer NOT NULL REFERENCES Meeting(CodMeeting),
3. UserID integer NOT NULL REFERENCES Partecipante(UserID)
4. );

### 4.1.10 Definizione della tabella **Ambito**

1. CREATE TABLE Ambito (
2. Nome enum\_ambito PRIMARY KEY
3. );

### 4.1.11 Definizione della tabella **ProgAmbito**

1. CREATE TABLE ProgAmbito (
2. CodProgetto integer NOT NULL REFERENCES Progetto(CodProgetto),
3. Nome enum\_ambito NOT NULL REFERENCES Ambito(Nome)
4. );
6. CREATE OR REPLACE FUNCTION storico\_progetti()
7. Returns Trigger as
8. $FineProgetto$
9. BEGIN
10. IF new.terminato=TRUE THEN
11. INSERT INTO ProgRealizzato
12. VALUES (old.codprogetto, old.tipologia);
14. DELETE FROM Progetto
15. WHERE CodProgetto=old.codprogetto;
16. END IF;
17. RETURN NEW;
18. END;
19. $FineProgetto$ LANGUAGE PLPGSQL;
21. CREATE TRIGGER FineProgetto
22. AFTER UPDATE ON Progetto
23. FOR EACH ROW
24. EXECUTE PROCEDURE storico\_progetti();
25. CREATE OR REPLACE FUNCTION controllo\_partecipante\_meeting()
26. Returns Trigger as
27. $PartecipanteMeeting$
28. DECLARE partecipanti integer;
30. BEGIN
31. SELECT userid INTO partecipanti
32. FROM CompMeeting AS C NATURAL JOIN Meeting AS M
33. WHERE orainizio IS NOT NULL AND orafine IS NULL;
35. IF new.userid IN (partecipanti) THEN
36. RAISE 'Un dipendente può partecipare a solo un meeting alla volta';
37. RETURN NULL;
38. END IF;
39. RETURN new;
40. END;
41. $PartecipanteMeeting$ LANGUAGE PLPGSQL;
43. CREATE TRIGGER PartecipanteMeeting
44. BEFORE INSERT ON Compmeeting
45. FOR EACH ROW
46. EXECUTE PROCEDURE controllo\_partecipante\_meeting();
47. CREATE OR REPLACE FUNCTION Controllo\_luogo()
48. Returns Trigger as
49. $LuogoOccupato$
50. DECLARE n\_riunioni integer;
52. BEGIN
53. SELECT COUNT(\*) INTO n\_riunioni
54. FROM MEETING AS M
55. WHERE luogo=new.luogo AND DataRiunione=new.DataRiunione AND
56. (new.OraInizio >= OraInizio AND new.OraInizio <= OraFine);
58. IF (n\_riunioni > 0) THEN
59. RAISE 'Errore, il luogo è già occupato da una riunione';
60. RETURN NULL;
61. END IF;
62. RETURN NEW;
63. END;
64. $LuogoOccupato$ LANGUAGE PLPGSQL;
66. CREATE TRIGGER LuogoOccupato
67. BEFORE INSERT ON Meeting
68. FOR EACH ROW
69. EXECUTE PROCEDURE Controllo\_Luogo();
70. CREATE OR REPLACE FUNCTION check\_meeting\_permesso()
71. Returns Trigger as
72. $MeetingNonPermesso$
73. DECLARE utenti\_liberi REFCURSOR;
74. DECLARE progettista\_libero integer;
76. BEGIN
77. OPEN utenti\_liberi FOR (SELECT UserID FROM Partecipante WHERE CodProgetto IS NULL);
78. LOOP
79. FETCH utenti\_liberi INTO progettista\_libero;
80. IF new.UserID = progettista\_libero THEN
81. RAISE 'Impossibile partecipare a questo meeting: il progettista non ha attualmente alcun progetto a carico';
82. RETURN NULL;
83. END IF;
84. RETURN NEW;
85. END LOOP;
86. CLOSE utenti\_liberi;
88. END;
89. $MeetingNonPermesso$ LANGUAGE PLPGSQL;
91. CREATE TRIGGER MeetingNonPermesso
92. BEFORE INSERT ON CompMeeting
93. FOR EACH ROW
94. EXECUTE PROCEDURE check\_meeting\_permesso();
95. CREATE OR REPLACE FUNCTION Check\_PM()
96. Returns Trigger as $ProjectManager$
97. DECLARE npm integer;
98. BEGIN
99. IF UPPER(new.ruolo) <> 'PROJECT MANAGER' THEN
100. return NEW;
101. ELSE
102. SELECT count(\*) INTO npm
103. FROM partecipante
104. WHERE UPPER(ruolo) = 'PROJECT MANAGER'
105. AND codprogetto=new.codprogetto;
107. if (npm > 0) then
108. raise 'Errore, un progetto non può avere più di un project manager';
109. return NULL;
110. end if;
111. return new;
112. END IF;
113. END;
114. $ProjectManager$ LANGUAGE PLPGSQL;
116. CREATE TRIGGER ProjectManager
117. BEFORE INSERT ON Partecipante
118. FOR EACH ROW
119. EXECUTE PROCEDURE Check\_PM();
120. CREATE OR REPLACE FUNCTION Check\_Progetto\_Mismatch()
121. Returns Trigger as
122. $ProgettoMismatch$
123. BEGIN
124. IF new.UserID IN (SELECT UserID
125. FROM Partecipante AS PA JOIN Meeting AS ME ON PA.CodProgetto=ME.CodProgetto
126. WHERE new.CodMeeting IN (SELECT CodMeeting
127. FROM Meeting
128. WHERE CodProgetto <> PA.CodProgetto)) THEN
129. RAISE 'Il progettista può partecipare solo a meeting su progetti a cui partecipa';
130. RETURN NULL;
131. END IF;
132. RETURN NEW;
133. END;
134. $ProgettoMismatch$ LANGUAGE PLPGSQL;
136. CREATE TRIGGER ProgettoMismatch
137. BEFORE INSERT ON CompMeeting
138. FOR EACH ROW
139. EXECUTE PROCEDURE Check\_Progetto\_Mismatch();
140. CREATE OR REPLACE FUNCTION Check\_Valutazione()
141. Returns Trigger as
142. $ValutazioneAziendale$
143. BEGIN
144. IF old.valutazione IS NOT NULL THEN
145. RAISE 'Un partecipante non può avere più di una valutazione per progetto';
146. RETURN NULL;
147. ELSE
148. UPDATE Partecipante
149. SET valutazione=new.valutazione;
150. END IF;
151. RETURN NEW;
152. END;$ValutazioneAziendale$ LANGUAGE PLPGSQL;
154. CREATE TRIGGER ValutazioneAziendale
155. BEFORE UPDATE OF Valutazione ON Partecipante
156. FOR EACH ROW
157. EXECUTE PROCEDURE Check\_Valutazione();
159. CREATE OR REPLACE FUNCTION Min\_Partecipanti\_Meeting()
160. Returns Trigger as
161. $ComposizioneMeeting$
162. DECLARE N\_Partecipanti integer;
163. DECLARE N\_Progettisti integer;
165. BEGIN
166. IF old.OraInizio IS NULL THEN
167. SELECT COUNT(UserID) INTO N\_Partecipanti
168. FROM CompMeeting
169. WHERE CodMeeting = new.CodMeeting;
171. SELECT COUNT(UserID) INTO N\_Progettisti
172. FROM PartecipanteProg
173. WHERE CodProg = new.CodMeeting;
175. IF N\_Partecipanti < (1/3 \* (N\_Progettisti)) THEN
176. RAISE 'Non è possibile avviare un meeting su un progetto con un numero di partecipanti minore ad 1/3 del numero totale di partecipanti al progetto in questione.';
177. RETURN NULL;
178. END IF;
179. RETURN NEW;
180. END IF;
181. END;
182. $ComposizioneMeeting$ LANGUAGE PLPGSQL;
184. CREATE TRIGGER Meeting
185. BEFORE UPDATE OF Orainizio ON Meeting
186. FOR EACH ROW
187. EXECUTE PROCEDURE Min\_Partecipanti\_Meeting();
189. CREATE OR REPLACE FUNCTION Meeting\_Senza\_PM()
190. Returns Trigger as
191. $Meeting\_Senza\_Manager$
192. DECLARE Project\_Manager integer;
193. BEGIN
194. IF old.OraInizio IS NULL THEN
195. SELECT UserID INTO Project\_Manager
196. FROM Partecipante
197. WHERE Ruolo = 'Project Manager' AND CodProgetto = new.CodProgetto;
198. IF Project\_Manager NOT IN (SELECT UserId FROM CompMeeting WHERE CodMeeting = new.CodMeeting) THEN
199. RAISE 'Non è possibile avviare un meeting su un progetto senza il relativo project manager.';
200. RETURN NULL;
201. END IF;
202. RETURN NEW;
203. END IF;
204. END;
205. $Meeting\_Senza\_Manager$ LANGUAGE PLPGSQL;
206. CREATE TRIGGER Meeting\_Senza\_Manager
207. BEFORE UPDATE OF Orainizio ON Meeting
208. FOR EACH ROW
209. EXECUTE PROCEDURE Meeting\_Senza\_PM();