Progetto Di Basi Di Dati 2022/2023
Sistema Di Gestione Di Personale E Progetti All'Interno Di Un'Azienda

Napolano
Emilia
N86004758

Liberti
Simone
N86004252

Novembre 2022

Indice

Capitolo 1

Analisi del progetto e requisiti individuati

1.1 Dipendenti dell'azienda

Per ogni dipendente si assegnerà una categoria basata sul numero di anni di servizio. Ogni dipendente appartiene a una sola fra quattro categorie:

- Dipendente Junior
- Dipendente Middle
- Dipendente Senior
- Dipendente Dirigente

Ogni dipendente può essere identificato con un nome, un cognome e un identificativo unico nel sistema (codice fiscale). Per ogni dipendente può essere specificato una mansione e un ufficio.

1.2 Passaggio di ruolo

Le categorie sono assegnate ai dipendenti in base all'anzianità, rispettivamente:

- Junior: meno di 3 anni
- Middle: compreso tra 3 e 7 anni
- Senior: più di 7 anni

L'unica categoria che non richiede anzianità è la categoria Dirigente, che può essere raggiunta solo in base alle proprie capacità. Di conseguenza sarà necessario tracciare i dati di carriera di ogni dipendente, memorizzando la prima data di assunzione e il numero di anni di servizio attraverso un'entità separata.

1.3 Gestione dei laboratori

I laboratori sono formati da gruppi di dipendenti di qualsiasi categoria che lavorano a specifici topic di un determinato progetto. Un laboratorio può essere gestito solo da un dipendente senior, che avrà titolo di responsabile scientifico. Ad ogni progetto saranno assegnati al più 3 laboratori.

1.4 Gestione dei progetti

Un progetto è identificato da un CUP (Codice Unico Progetto) e da un nome (unico nel sistema). Ogni progetto può essere gestito solo da un dipendente Dirigente e avrà, come per i laboratori, un referente scientifico, che potrà essere solo un dipendente senior.

1.5 Requisiti aggiuntivi

Come requisiti aggiuntivi sono stati identificati:

- Tracciamento dei contratti
- Collaborazione tra più aziende

Un contratto può essere caratterizzato da un tipo e una durata temporale, oltre a tenere traccia della data della firma e del salario.
Più aziende possono collaborare a uno stesso progetto.
Un contratto a tempo determinato viene rinnovato automaticamente ogni 3 anni. Un dipendente puo' avere un aumento di salario anche senza passaggio di ruolo o senza aver firmato un nuovo contratto.

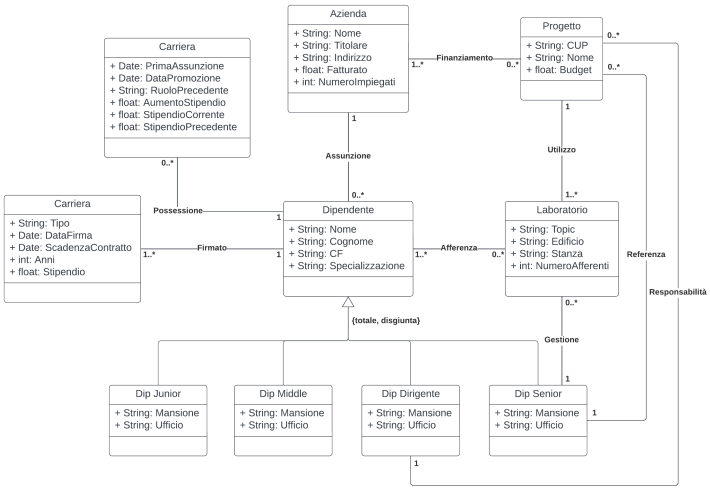
Capitolo 2

Progettazione concettuale

2.1 Introduzione

In questa sezione verrà introdotta la progettazione concettuale dell'applicativo. Partendo da un primo diagramma di classe in UML. Dal risultato dell'analisi dei requisiti che devono essere soddisfatti si arriverà a uno schema concettuale ristrutturato. Saranno evidenziati i concetti rilevanti ai fini della rappresentazione dei dati e le relazioni che intercorrono tra di esse.

2.2 Class Diagram



2.3 Ristrutturazione del Class Diagram

Al fine di rendere il class diagram idoneo alla traduzione logica e di migliorare l'efficienza dell'implementazione si procede alla ristrutturazione dello stesso.

2.3.1 Analisi delle ridondanze

Sono ora anlizzate le ridondanze trovate all'interno del Class Diagram precedente:

- Il numero degli impiegati può essere ricavato attraverso la relazione assunzione,
- Il numero di anni di contratto può essere ricavato sottraendo alla data di scadenza la data di assunzione,
- Il numero di afferenti di un laboratorio può essere ricavato contando i dipendenti dalla relazione Afferenza.

Da ricordare che la rimozione delle ridondanze è specifica alla progettazione concettuale: in fase di implementazione potrebbe rivelarsi conveniente introdurre alcuni attributi ridondanti al fine di facilitare le query (un esempio sono StipendioPrecedente e StipendioCorrente).

2.3.2 Analisi degli identificativi

Per facilitare la traduzione verso lo schema logico e per il recupero efficiente dei dati si inseriscono gli identificativi IDContratto e IDCarriera, rispettivamente alle entità Contratto e Carriera. Per identificare univocamente i dipendenti sarà considerato unico il codice fiscale. Per l'azienda sono presi come identificativi il nome e la via, mentre per il progetto sarà utilizzato il CUP. Infine per il laboratorio si possono considerare univoci nel sistema tutti i suoi attributi, ovvero "Topic, Edificio, Stanza".

2.3.3 Rimozione degli attributi multivalore

Il nome e il cognome del titolare dell'azienda saranno considerati come una stringa unica.

2.3.4 Rimozione degli attributi composti

Nell'entità Azienda l'attributo Indirizzo sarà estratto in modo da delineare gli attributi: "Via, Civico, CAP".

2.3.5 Partizione/Accorpamento delle associazioni

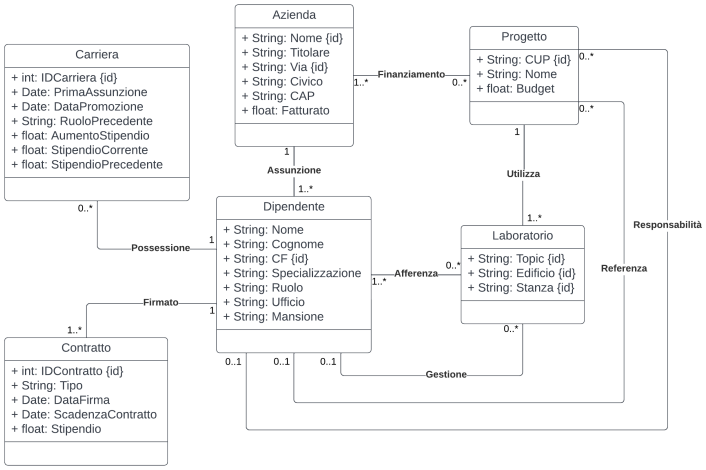
Non vengono eseguiti accorpamenti o partizioni delle associazioni.

2.3.6 Rimozione delle gerarchie

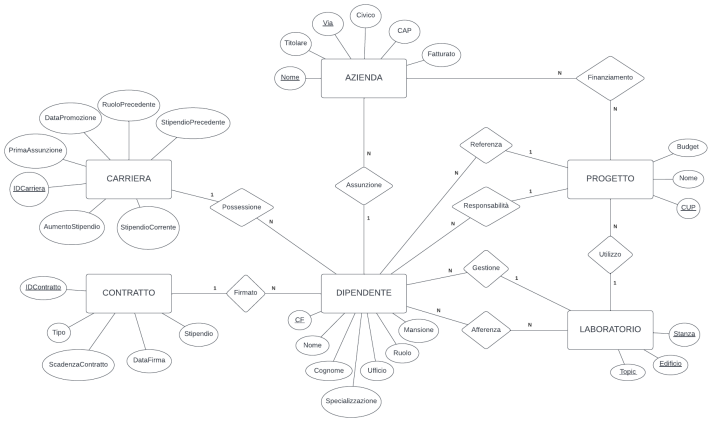
La disgiunzione totale tra la superclasse Dipendente e le sottoclassi Dipendente Junior, Dipendente Middle, Dipendente Senior e Dipendente Dirigente viene eliminata accorpando le sottoclassi nella superclasse: sarà necessario inserire un attributo "Ruolo" per controllare la categoria del dipendente.
Sarà anche necessario modificare le molteplicità delle associazioni tra la classe Dipendente e le classi Progetto e Laboratorio.

2.4 Diagrammi ristrutturati

2.4.1 Class Diagram



2.4.2 Diagramma ER



Capitolo 3

Dizionari

3.1 Introduzione

In questo capitolo verranno analizzate le singole entità, le loro associazioni ed eventuali vincoli e di seguito riportato un dizionario contenente le loro descrizioni e caratteristiche.

3.2 Dizionario delle classi

Tabella 3.1: Dizionario delle classi.

Classe	Descrizione	Attributi
Azienda	Entità descrittrice di un tipo generico di Azienda	**Nome** (String): Nome dell'azienda, identifica univocamente un'azienda. **Titolare** (String): Nome del titolare dell'azienda. **Via** (String): Via della sede dell'azienda, identifica univocamente un'azienda. **Civico** (String): Civico della sede dell'azienda. **CAP** (String): Codice di avviamento postale. **Fatturato** (float): Fatturato dell'azienda.

Continua nella pagina successiva

Tabella 3.1 Dizionario delle classi

Classe	Descrizione	Attributi
Dipendente	Entità descrittrice di un profilo generico di un dipendente aziendale	**Nome** (String): Nome del dipendente. **Cognome** (String): Cognome del dipendente. **CF** (String): Codice Fiscale del dipendente, identifica univocamente un dipendente. **Specializzazione** (String): Laurea del dipendente (se conseguita). **Ruolo** (String): Categoria di appartenenza del dipendente. **Ufficio** (String): Ufficio dove lavora il dipendente. **Mansione** (String): Lavoro svolto abitualmente dal dipendente.
Contratto	Entità descrittrice di un contratto firmato da un dipendente per una certa azienda	**IDContratto** (int): identifica univocamente un contratto. **Tipo** (String): Tipo di contratto firmato. **DataFirma** (Date): Data in cui il dipendente firma il contratto. **ScadenzaContratto** (Date): Data in cui scade il contratto (se è a tempo determinato). **Stipendio** (float): Stipendio accordato al momento della firma del contratto.

Continua nella pagina successiva

Tabella 3.1 Dizionario delle classi

Classe	Descrizione	Attributi
Carriera	Entità descrittrice della carriera attuale o pregressa di un dipendente.	**IDCarriera** (int): identifica univocamente una carriera **PrimaAssunzione** (Date): Data di prima assunzione del dipendente. **DataPromozione** (Date): Data della promozione del dipendente. **RuoloPrecedente** (String): Ultimo ruolo ricoperto dal dipendente prima della promozione. **AumentoStipendio** (float): Aumento stipendio in percentuale. **StipendioCorrente** (float): Stipendio aumentato a seguito dell'aumento di stipendio. **StipendioPrecedente** (float): Stipendio prima dell'aumento.
Progetto	Entità descrittrice di un progetto finanziato da un'azienda	**CUP** (String): Codice Unico Progetto, identifica univocamente un progetto. **Nome** (String): Nome del progetto. **Budget** (float): Budget del progetto.
Laboratorio	Entità descrittrice di un laboratorio.	**Topic** (String): Argomento principale del laboratorio, identifica univocamente un laboratorio. **Edificio** (String): Edificio in cui è situato il laboratorio, identifica univocamente un laboratorio. **Stanza** (String): Stanza in cui è situato il laboratorio, identifica univocamente un laboratorio.

3.3 Dizionario delle associazioni

Tabella 3.2: Dizionario delle associazioni.

Nome	Descrizione	Classi coinvolte
Finanziamento	Esprime la relazione tra l'azienda e il progetto. Un'azienda finanzia uno o più progetti, un progetto è finanziato da una o più aziende.	**Azienda [1...*]** ruolo **finanzia:** indica la/le aziende che finanziano un progetto. **Progetto [0...*]** ruolo **è finanziato da:** indica progetti che sono finaziati da una o più aziende.
Assunzione	Esprime la relazione tra l'azienda e il dipendente. Un'azienda assume uno o più dipendenti, un dipendente è assunto da un'azienda.	**Azienda [1]** ruolo **assume:** indica l'azienda che assume un dipendente. **Dipendente [1...*]** ruolo **è assunto da:** indica dipendenti che sono assunti da un'azienda.
Possessione	Esprime la relazione tra dipendente e carriera. Un dipendente possiede una o più carriere, una carriera è posseduta da un solo dipendente.	**Dipendente [1]** ruolo **possiede:** indica il dipendente che possiede una carriera. **Carriera[0...*]** ruolo **è posseduta da:** indica la carriera attuale o le carriere pregresse possedute da un dipendente.
Firmato	Esprime la relazione tra dipendente e contratto. Un dipendente firma uno o più contratti, un contratto è firmato da un solo dipendente.	**Dipendente [1]** ruolo **firma:** indica il dipendente che firma un contratto. **Contratto [1...*]** ruolo **è firmato da:** indica il/i contratti firmati da un dipendente.

Continua nella pagina successiva

Tabella 3.2 Dizionario delle associazioni

Nome	Descrizione	Classi coinvolte
Gestione	Esprime la relazione tra dipendente senior e laboratorio. Un dipendente gestisce più laboratori, un laboratorio è gestito da un solo dipendente.	**Dipendente [0...1]** ruolo **gestisce:** indica il dipendente che gestisce un laboratorio. **Laboatorio [0...*]** ruolo **è gestito da:** indica il laboratorio che è gestito da un dipendente.
Referenza	Esprime la relazione tra dipendente senior e progetto. Un dipendente è referente di uno o più progetti, un progetto è referenziato da un solo dipendente.	**Dipendente [0...1]** ruolo **è referente di:** indica il dipendente che è referente scientifico di un progetto. **Progetto [0...*]** ruolo **è referenziato da:** indica il progetto che è referenziato da un dipendente.
Responsabilità	Esprime la relazione tra dirigente e progetto. Un dipendente è responsabile di uno o più progetti, un progetto ha come responsabile un solo dipendente.	**Dipendente [0...1]** ruolo **è responsabile di:** indica il dipendente che è responsabile di un progetto. **Progetto [0...*]** ruolo **ha come responsabile:** indica il progetto che ha come responsabile un dipendente.
Afferenza	Esprime la relazione tra dipendente e laboratorio. Un dipendente afferisce a uno o più laboratori, un laboratorio è afferito da uno o più dipendenti.	**Dipendente [1...*]** ruolo **afferisce:** indica il dipendente che afferisce a un laboratorio. **Laboratorio [0...*]** ruolo **è afferito da:** indica il laboratorio che è afferito da un dipendente.

Continua nella pagina successiva

Tabella 3.2 Dizionario delle associazioni

Nome	Descrizione	Classi coinvolte
Utilizzo	Esprime la relazione tra progetto e laboratorio. Un laboratorio è utilizzato da un solo progetto, un progetto utilizza uno o più laboatori.	**Laboratorio [1...*]** ruolo **è utilizzato da:** indica il laboratorio che è utilizzato da un progetto. **Progetto [1]** ruolo **utilizza:** indica il progetto che utilizza un laboratorio.

3.4 Dizionario dei vincoli

Tabella 3.3: Dizionario dei vincoli.

Nome Vincolo	Descrizione
Validità Nome Azienda	Il nome di un'azienda deve essere composto unicamente da caratteri alfanumerici.
Validità Nome Titolare	Il nome di un titolare deve essere composto unicamente da lettere.
Dominio ruoli	Il ruolo di un dipendente può esclusivamente essere uno solo tra "Junior", "Middle", "Senior", "Dirigente".
Anzianità	Il passaggio di ruolo da "Junior" a "Middle" deve avvenire solo dopo 3 anni di esperienza. Il passaggio da "Middle" a "Senior" deve avvenire dopo 7 anni di servizio.
Gestione Laboratorio	Un laboratorio può essere gestito solo da un dipendente "Senior".
Utilizzo Laboratori	Un progetto può utilizzare al più 3 laboratori.
Nome Progetto	Il nome di ogni progetto deve essere unico nel sistema.
Referenza Progetti	Un progetto può avere come referente solo un dipendente "Senior".
Responsabilità Progetti	Un progetto può avere come responsabile solo un dipendente "Dirigente".
Validità CF	Un Codice Fiscale deve contenere al massimo 16 caratteri alfanumerici.

Continua nella pagina successiva

Tabella 3.3 Dizionario dei vincoli

Nome Vincolo	Descrizione
Validità Data Contratto	La data di scadenza di un contratto non può essere precedente alla data di firma.
Validità Data Carriera	La data di promozione di un dipendente non può essere precedente alla data di prima assunzione.
Consistenza Prima Assunzione	La data di prima assunzione deve coincidere con la data di firma del primo contratto.
Dominio Aumento Stipendio	Il valore di aumento di stipendio deve essere compreso tra 0 e 1
Validità Nome e Cognome	Il nome e cognome di ogni dipendente devono contenere almeno un carattere e solo caratteri compresi tra A-Z o a-z.
Indeterminato senza scadenza	Se il contratto di un dipendente è a tempo indeterminato allora la scadenza del contratto deve essere null.

Capitolo 4

Progettazione Logica

In questo capitolo verrà trattato il passaggio dalla progettazione concettuale alla progettazione logica. Nello schema che seguirà le chiavi primarie saranno identificate con una sottolineatura e le chiavi esterne con un *.

4.1 Schema Logico

Contratto	(IDContratto, Tipo, DataFirma, Scadenza-Contratto, Stipendio, CF*) CF → Dipendente.CF
Dipendente	(Nome, Cognome, CF, Specializzazione, Ruolo, Ufficio, Mansione)
Laboratorio	(Topic,Edificio, Stanza)
Progetto	(CUP, Nome, Budget)
Utilizzo	(CUP*,Topic*,Edificio*,Stanza*) CUP→Progetto.CUP (Topic,Edificio,Stanza)→Laboratorio.(Topic,Edificio,Stanza)
Carriera	(IDCarriera, PrimaAssunzione, DataPromozione, RuoloPrecedente, AumentoStipendio, StipendioCorrente, StipendioPrecedente)

Possessione	(IDCarriera*, CF*) IDCarriera→Carriera.IDCarriera CF → Dipendente.CF
Azienda	(Nome, Titolare,Via, Civico, CAP, Fatturato)
Assunzione	(Nome*, Via*, CF*) Nome → Azienda.Nome Via → Azienda.Via CF → Dipendente.CF
Finanziamento	(Nome*, Via*,CUP*) Nome → Azienda.Nome Via → Azienda.Via CUP → Progetto.CUP
Afferenza	(CF*, Topic*, Edificio*, Stanza*) CF → Dipendente.CF (Topic,Edificio,Stanza) → Laboratorio.(Topic,Edificio,Stanza)
Responsabilità	(CF*, CUP*) CF → Dipendente.CF CUP → Progetto.CUP
Gestione	(CF*, Topic*, Edificio*, Stanza*) CF → Dipendente.CF (Topic,Edificio,Stanza) → Laboratorio.(Topic,Edificio,Stanza)
Referenza	(CF*, CUP*) CF → Dipendente.CF CUP → Progetto.CUP

4.2 Trigger e Procedure

4.2.1 Indeterminato senza scadenza

```
--Indeterminato senza scadenza: questa trigger function verifica che all'inserimento di un
Contratto, se il tipo è Indeterminato allora la scadenza deve essere null, altrimenti se il
tipo è Determinato, la scadenza non deve essere null.
CREATE OR REPLACE FUNCTION Chiamata_Trigger_Indeterminato_Senza_Scadenza() RETURNS TRIGGER AS
$$
BEGIN
IF ((new.Tipo='Indeterminato' AND new.ScadenzaContratto IS NULL) OR
(new.Tipo='Determinato' AND new.ScadenzaContratto IS NOT NULL)) THEN --ciò che va bene
RETURN NEW;
ELSIF ((new.Tipo='Determinato' AND new.ScadenzaContratto IS NULL) OR
(new.Tipo='Indeterminato' AND new.ScadenzaContratto IS NOT NULL)) THEN --ciò che non va bene
RAISE EXCEPTION 'Vincolo tipo-scadenza contratto non rispettato.';
RETURN NULL;
END IF;
END;
$$
language plpgsql;

CREATE OR REPLACE TRIGGER Trigger_Indeterminato_Senza_Scadenza BEFORE INSERT ON Contratto
FOR EACH ROW
WHEN(new.Tipo IS NOT NULL)
EXECUTE PROCEDURE Chiamata_Trigger_Indeterminato_Senza_Scadenza();
```

Esempio di cosa accade se si prova ad inserire un contratto non valido:

```
1  INSERT INTO CONTRATTO(Tipo,DataFirma,ScadenzaContratto,Stipendio,CF)
2  VALUES('Determinato','now',NULL,1200,'F123456789012345');
```

Data output | Messages | Notifications

```
ERROR: ERRORE:  Vincolo tipo-scadenza contratto non rispettato.
CONTEXT:  funzione PL/pgSQL chiamata_trigger_indeterminato_senza_scadenza() riga 6 a RAISE
```

4.2.2 Passaggio a indeterminato

```
--Questa funzione serve per modificare un contratto da tempo determinato a tempo indeterminato
CREATE OR REPLACE FUNCTION ToIndeterminate(CODICEF Dipendente.CF%TYPE) RETURNS BOOLEAN AS $$
DECLARE
LastContract Contratto.IDContratto%TYPE;
MaxFirma Contratto.DataFirma%TYPE;
MyScadenza Contratto.ScadenzaContratto%TYPE;
MyStip Carriera.StipendioCorrente%TYPE;
MaxPromozione Carriera.IDCARRIERA%TYPE;
BEGIN
--Prendiamo il contratto più recente del dipendente
SELECT MAX(C1.DataFirma) INTO MaxFirma
FROM CONTRATTO AS C1
WHERE C1.CF=CODICEF;
--Prendiamo l'id del contratto e la scadenza del contratto con firma più recente
SELECT C1.IDContratto,C1.ScadenzaContratto INTO LastContract, MyScadenza
FROM CONTRATTO AS C1
WHERE C1.CF=CODICEF AND C1.DATAFIRMA=MaxFirma
GROUP BY C1.IDContratto;
--Se il dipendente possiede già un contratto a tempo indeterminato non devo fare niente
IF(MyScadenza IS NULL) THEN
    RAISE NOTICE 'IL DIPENDENTE % POSSIEDE GIA UN CONTRATTO A TEMPO INDETERMINATO', CODICEF;
    RETURN FALSE;
ELSE
    --Prendo la data di promozione più recente del dipendente per poter ricavare lo stipendio
corrente da inserire nel nuovo contratto
    SELECT MAX(C1.IDCARRIERA) INTO MaxPromozione
    FROM CARRIERA AS C1 JOIN POSSESSIONE AS P1 ON C1.IDCARRIERA=P1.IDCARRIERA
    WHERE P1.CF=CODICEF;
    --Prendo lo stipendio corrente dall'ultima promozione
    SELECT C1.StipendioCorrente INTO MyStip
    FROM (DIPENDENTE AS D JOIN POSSESSIONE AS P1 ON D.CF=P1.CF) JOIN CARRIERA AS C1 ON
C1.IDCARRIERA=P1.IDCARRIERA
    WHERE D.CF=CODICEF AND C1.IDCARRIERA=MaxPromozione;
    --Inserisco il nuovo contratto
    INSERT INTO CONTRATTO(Tipo,DataFirma,ScadenzaContratto,Stipendio,CF)
    VALUES('Indeterminato','now',NULL,MyStip,CODICEF);
    RETURN TRUE;
END IF;
END;
$$
language plpgsql;
```

Esempio di cosa accade se si prova a modificare un contratto da determinato a indeterminato:

```
1  select ToIndeterminate('1234567890ABCDEF');
```

Data output | Messages | Notifications

```
Successfully run. Total query runtime: 941 msec.
1 rows affected.
```

Esempio di cosa accade se si prova a modificare un contratto a indeterminato se è già indeterminato.

```
1  select ToIndeterminate('1234567890ABCDEF');
```

Data output | Messages | Notifications

```
NOTIFICA:  IL DIPENDENTE 1234567890ABCDEF POSSIEDE GIA UN CONTRATTO A TEMPO INDETERMINATO

Successfully run. Total query runtime: 59 msec.
1 rows affected.
```

4.2.3 Rinnovo contratto scaduto

```
--ScadenzaContratto: queste funzioni servono per rinnovare i contratti a tempo Determinato
qual'ora fossero scaduti
CREATE OR REPLACE FUNCTION IsObsolete(CODICEF IN Dipendente.CF%TYPE) RETURNS BOOLEAN AS $$
DECLARE
LastContract Contratto.IDContratto%TYPE;
MaxFirma Contratto.DataFirma%TYPE;
MyScadenza Contratto.ScadenzaContratto%TYPE;
BEGIN
--Prendiamo il contratto più recente del dipendente
SELECT MAX(C1.DataFirma) INTO MaxFirma
FROM CONTRATTO AS C1
WHERE C1.CF=CODICEF;
--Prendiamo l'id del contratto e la scadenza del contratto con firma più recente
SELECT C1.IDContratto,C1.ScadenzaContratto INTO LastContract, MyScadenza
FROM CONTRATTO AS C1
WHERE C1.CF=CODICEF AND C1.DATAFIRMA=MaxFirma
GROUP BY C1.IDContratto;
--Se il contratto è scaduto allora IsObsolete è vera
IF('now'>MyScadenza) THEN
RETURN TRUE;
END IF;
RETURN FALSE;
END;
$$
language plpgsql;

--Rinnovo del contratto
CREATE OR REPLACE FUNCTION Renewal_Contract(CODICEF IN Dipendente.CF%TYPE) RETURNS BOOLEAN AS
$$
DECLARE
MyType Dipendente.Ruolo%TYPE;
MyStip Carriera.StipendioCorrente%TYPE;
MaxPromozione Carriera.IDCARRIERA%TYPE;
BEGIN
--Prendo la data di promozione più recente del dipendente per poter icavare lo stipendio
corrente da inserire nel nuovo contratto
SELECT MAX(C1.IDCARRIERA) INTO MaxPromozione
FROM CARRIERA AS C1 JOIN POSSESSIONE AS P1 ON C1.IDCARRIERA=P1.IDCARRIERA
WHERE P1.CF=CODICEF;
--Prendo lo stipendio corrente dall'ultima promozione
SELECT C1.StipendioCorrente INTO MyStip
FROM (DIPENDENTE AS D JOIN POSSESSIONE AS P1 ON D.CF=P1.CF) JOIN CARRIERA AS C1 ON
C1.IDCARRIERA=P1.IDCARRIERA
WHERE D.CF=CODICEF AND C1.IDCARRIERA=MaxPromozione;
--Se IsObsolete è vera allora inserisco un nuovo contratto, per scelta aziendale un contratto
a tempo determinato scade ogni 3 anni
IF(IsObsolete(CODICEF)) THEN
INSERT INTO CONTRATTO(tipo,DataFirma,ScadenzaContratto,Stipendio,CF)
VALUES('Determinato','now',current_date+1095, MyStip, CODICEF);
RAISE NOTICE 'Rinnovo contratto di % effettuato correttamente.', CODICEF;
RETURN TRUE;
END IF;
RAISE NOTICE 'Non occorre rinnovare il contratto.';
RETURN FALSE;
END;
$$
language plpgsql;
```

```
--Questa funzione verifica se tutti i dipendenti di una certa azienda sono idonei al rinnovo
contratto
CREATE OR REPLACE PROCEDURE Check_For_Global_Renewal(nome IN Azienda.NOME%TYPE, via IN
Azienda.VIA%TYPE) AS $$
DECLARE
--Scrivo il comando per prelevare ogni codice fiscale di dipendenti di una certa azienda
comandoSQL VARCHAR(1000):='SELECT A.CF
                           FROM ASSUNZIONE AS A
                           WHERE A.VIA=$1 AND A.NOME=$2';
cursore REFCURSOR;
CODICEF Dipendente.CF%TYPE;
BEGIN
--apro il cursore utilizzando i parametri di ingresso
OPEN cursore FOR EXECUTE comandoSQL using via, nome;
LOOP
--prelevo il codice fiscale fin quando finiscono i dipendenti
FETCH cursore INTO CODICEF;
EXIT WHEN NOT FOUND;
--chiamo la funzione Check_For_Promotion
perform Renewal_Contract(CODICEF);
END LOOP;
CLOSE cursore;
END;
$$
language plpgsql;
```

Esempio di cosa accade se rinnovo tutti i contratti scaduti di una certa azienda:

```
1  call check_for_global_renewal('Walmart','Ave Hostos');
```

Data output | Messages | Notifications

```
NOTIFICA:  Rinnovo contratto di F123456789012345 effettuato correttamente.
NOTIFICA:  Rinnovo contratto di F234567890123456 effettuato correttamente.
NOTIFICA:  Rinnovo contratto di F345678901234567 effettuato correttamente.
NOTIFICA:  Rinnovo contratto di F456789012345678 effettuato correttamente.
NOTIFICA:  Rinnovo contratto di F567890123456789 effettuato correttamente.
NOTIFICA:  Rinnovo contratto di F678901234567890 effettuato correttamente.
```

4.2.4 Assunzione dei dipendenti e firma del contratto

```
--Generazione automatica del contratto al momento dell'assunzione dei dipendenti
CREATE OR REPLACE FUNCTION Generate_Contract() RETURNS TRIGGER AS $$
DECLARE
MyID Carriera.IDCARRIERA%TYPE;
MyRuolo Dipendente.Ruolo%TYPE;
BEGIN
--Prelevo il ruolo del dipendente che sto assumendo
SELECT D.Ruolo INTO MyRuolo
FROM DIPENDENTE AS D
WHERE New.CF=D.CF;
--Inserisco il nuovo contratto e la carriera del dipendente in base al ruolo,
--per motivi di prova dei trigger correttamente decido di inserire
--il primo contratto il 2016-01-01 come se fosse la mia data corrente,
--mentre la scadenza è la mia data corrente
IF(MyRuolo='Junior') THEN
INSERT INTO CONTRATTO(tipo,datafirma,scadenzacontratto,stipendio,cf)
VALUES ('Determinato','2016-01-01',current_date,1100,New.CF);
INSERT INTO CARRIERA(primaassunzione,datapromozione,ruoloprecedente,
aumentostipendio,stipendiocorrente,stipendioprecedente)
VALUES ('2016-01-01',NULL,NULL,NULL,1100,NULL);

ELSIF (MyRuolo='Middle') THEN
INSERT INTO CONTRATTO(tipo,datafirma,scadenzacontratto,stipendio,cf)
VALUES ('Determinato','2016-01-01',current_date,1210,New.CF);
INSERT INTO CARRIERA(primaassunzione,datapromozione,ruoloprecedente,
aumentostipendio,stipendiocorrente,stipendioprecedente)
VALUES ('2016-01-01',NULL,'Junior',NULL,1210,NULL);

ELSIF (MyRuolo='Senior') THEN
INSERT INTO CONTRATTO(tipo,datafirma,scadenzacontratto,stipendio,cf)
VALUES ('Determinato','2016-01-01',current_date,1452,New.CF);
INSERT INTO CARRIERA(primaassunzione,datapromozione,ruoloprecedente,
aumentostipendio,stipendiocorrente,stipendioprecedente)
VALUES ('2016-01-01',NULL,'Middle',NULL,1452,NULL);

ELSIF (MyRuolo='Dirigente') THEN
INSERT INTO CONTRATTO(tipo,datafirma,scadenzacontratto,stipendio,cf)
VALUES ('Determinato','2016-01-01',current_date,1815,New.CF);
INSERT INTO CARRIERA(primaassunzione,datapromozione,ruoloprecedente,
aumentostipendio,stipendiocorrente,stipendioprecedente)
VALUES ('2016-01-01',NULL,NULL,NULL,1815,NULL);

END IF;
--Prendiamo l'id dell'ultima carriera inserita
SELECT MAX(C1.IDCARRIERA) INTO MyID
FROM CARRIERA AS C1;
--Inserisco quest'ultima carriera nella tupla di possessione del dipendente
INSERT INTO POSSESSIONE(IDCarriera,CF)
VALUES (MyID,New.CF);
RETURN NEW;
END;
$$
language plpgsql;

CREATE OR REPLACE TRIGGER Trigger_Generate_Contract AFTER INSERT ON ASSUNZIONE
FOR EACH ROW
WHEN(New.CF IS NOT NULL)
EXECUTE PROCEDURE Generate_Contract();
```

4.2.5 Utilizzo laboratori

```
--Utilizzo Laboratori: questa trigger function serve nel momento in cui
--ad un progetto vengono assegnati più di 3 laboratori, questa situazione genera un problema
CREATE OR REPLACE FUNCTION Chiamata_Trigger_Utilizzo_Laboratori() RETURNS TRIGGER AS $$
DECLARE
numero_laboratori INTEGER;
BEGIN
--Conto il numero di laboratori che sono utilizzati dal progetto
SELECT COUNT(U.TOPIC) INTO numero_laboratori
FROM UTILIZZO AS U
WHERE New.CUP=U.CUP;
--Se il numero di laboratori è già 3 allora non va bene
IF (numero_laboratori=3) THEN
RAISE EXCEPTION 'Il progetto lavora già su 3 laboratori.';
RETURN NULL;
ELSIF (numero_laboratori<3) THEN
return new;
END IF;
END;
$$
language plpgsql;

CREATE OR REPLACE TRIGGER Trigger_Utilizzo_Laboratori BEFORE INSERT ON UTILIZZO
FOR EACH ROW
EXECUTE PROCEDURE Chiamata_Trigger_Utilizzo_Laboratori();
```

Esempio di cosa accade se un progetto tenta di utilizzare più di 3 laboratori:

```
1  INSERT INTO UTILIZZO(CUP,Topic,Edificio,Stanza)
2  VALUES('CUP_1','Ingegneria Informatica','Palazzo dell''Innovazione','D2')
```

Data output | Messages | Notifications

```
ERROR: ERRORE:  Il progetto lavora già su 3 laboratori.
CONTEXT:  funzione PL/pgSQL chiamata_trigger_utilizzo_laboratori() riga 11 a RAISE
```

4.2.6 Passaggio a nuovo ruolo

```
--Conteggio anni di lavoro: conto quanti anni ha lavorato il dipendente nell'azienda
CREATE OR REPLACE FUNCTION Anni_Lavoro(CODICEF IN Dipendente.CF%TYPE) RETURNS INTEGER AS $$
DECLARE
DataPrimaAssunzione Carriera.PrimaAssunzione%TYPE;
anni INTEGER;
BEGIN
--Prelevo la data di prima assunzione del dipendente
SELECT C1.PrimaAssunzione INTO DataPrimaAssunzione
FROM (DIPENDENTE AS D JOIN POSSESSIONE AS P1 ON D.CF=P1.CF) JOIN CARRIERA AS C1 ON
P1.IDCARRIERA=C1.IDCARRIERA
WHERE D.CF=CODICEF;
--Calcolo gli anni di lavoro
anni:=EXTRACT(YEARS FROM age(CURRENT_DATE, DataPrimaAssunzione));
RAISE NOTICE 'Il dipendente % ha lavorato per % anni' , CODICEF, anni;
return anni;
END;
$$
language plpgsql;
```

```
--Anzianità: questa funzione verifica in quale range di anni il dipendente
-- sta lavorando e in base a quel range effettua il passaggio adeguato
CREATE OR REPLACE FUNCTION Anzianita(CODICEF IN Dipendente.CF%TYPE) RETURNS BOOLEAN AS $$
DECLARE
passaggio BOOLEAN:=false;
anni_Carriera INTEGER;
MyRuolo Dipendente.Ruolo%TYPE;
BEGIN
--Prelevo il ruolo del dipendente
SELECT D.RUOLO INTO MyRuolo
FROM DIPENDENTE AS D
WHERE D.CF=CODICEF;
--Calcolo gli anni di lavoro di quel dipendente attraverso la funzione Anni_Lavoro
SELECT Anni_Lavoro(CODICEF) INTO anni_Carriera;
--Se gli anni sono compresi tra 3 e 7 escluso allora effettuo il passaggio a middle
IF (anni_Carriera>=3 AND anni_Carriera<7) THEN
--Se il ruolo è già middle non c'è bisogno di effettuare il passaggio
    IF(MyRuolo = 'Junior') THEN
    passaggio=passaggio_middle(CODICEF);
    END IF;
--Se gli anni sono maggiori di 7 compreso allora effettuo il passaggio a senior
ELSEIF (anni_Carriera>=7) THEN
--Se il ruolo è già senior non c'è bisogno di effettuale il passaggio
    IF(MyRuolo = 'Middle') THEN
    passaggio=passaggio_senior(CODICEF);
    --Se non ho avuto per qualche errore il passaggio di promozione prima dei 7 anni
    --e sono Junior ora effettuo il passaggio prima a middle e poi a senior
    ELSIF(MyRuolo = 'Junior') THEN
    passaggio=passaggio_middle(CODICEF);
    passaggio=passaggio_senior(CODICEF);
    END IF;
END IF;
return passaggio;
END;
$$
language plpgsql;
```

```
--Passaggio a Middle
CREATE OR REPLACE FUNCTION passaggio_middle(CODICEF IN Dipendente.CF%TYPE) RETURNS BOOLEAN AS
$$
DECLARE
BEGIN
--Richiamo la procedura chiama_aumento_stipendio
call chiama_aumento_stipendio(CODICEF,0.1); --E' una decisione aziendale che per il passaggio
a Middle lo stipendio viene aumentato del 10%
--Faccio l'update del ruolo del dipendente
UPDATE DIPENDENTE AS D
SET RUOLO='Middle'
WHERE D.CF=CODICEF;
return true;
END;
$$
language plpgsql;
-----------------------------------
--Passaggio a Senior
CREATE OR REPLACE FUNCTION passaggio_senior(CODICEF IN Dipendente.CF%TYPE) RETURNS BOOLEAN AS
$$
DECLARE
BEGIN
--Richiamo la procedura chiama_aumento_stipendio
call chiama_aumento_stipendio(CODICEF,0.2); --E' una decisione aziendale che per il passaggio
a Senior lo stipendio viene aumentato del 20%
--Faccio l'update del ruolo del dipendente
UPDATE DIPENDENTE AS D
SET RUOLO='Senior'
WHERE D.CF=CODICEF;
return true;
END;
$$
language plpgsql;

--PassaggioDirigente: questa funzione serve per effettuare il passaggio dal ruolo attuale a
Dirigente
CREATE OR REPLACE FUNCTION passaggio_Dirigente(CODICEF IN Dipendente.CF%TYPE) RETURNS BOOLEAN
AS $$
DECLARE
MyID Carriera.IDCarriera%TYPE;
UltimaPromozione Carriera.IDCarriera%TYPE;
MyStip carriera.StipendioCorrente%TYPE;
PrimaA carriera.PrimaAssunzione%TYPE;
ora TIMESTAMP;
BEGIN
--Richiamo la procedura chiama_aumento_stipendio
call chiama_aumento_stipendio(CODICEF,0.25); --E' una decisione aziendale che per il passaggio
a Dirigente lo stipendio viene aumentato del 25%
--Faccio l'update del ruolo del dipendente
UPDATE DIPENDENTE AS D
SET RUOLO='Dirigente'
WHERE D.CF=CODICEF;
return true;
END;
$$
language plpgsql;
```

Esempio di cosa accade se effettuo un passaggio a Dirigente di un dipendente:

```
1  select Passaggio_Dirigente('F012345678901234');
```

Data output | Messages | Notifications

```
NOTIFICA:  Aumento lo stipendio del dipendente F012345678901234 in percentuale 0.25
NOTIFICA:  Lo stipendio è stato aumentato.
```

8	Francesca	Viola	F012345678901234	Progettazione	Dirigente	Progettazione	Settore Progettazione
51	745	2016-01-01 00:00:00	2023-01-16 15:36:33.724974	Senior	0.25	1815	1452

```
--Aumento dello stipendio: questa funzione inserisce una nuova carriera al
--dipendente e effettua opportuni calcoli in base al valore di aumento stipendio
CREATE OR REPLACE FUNCTION Aumento_Stipendio(CODICEF IN Dipendente.CF%TYPE, valore IN float)
RETURNS BOOLEAN AS $$
DECLARE
MyStip carriera.StipendioCorrente%TYPE;
MaxPromozione carriera.IDCARRIERA%TYPE;
PrimaA carriera.PrimaAssunzione%TYPE;
MaximumC carriera.IDCarriera%TYPE;
ora TIMESTAMP;
BEGIN
--Prelevo l'ultima promozione
SELECT MAX(C1.IDCARRIERA) INTO MaxPromozione
FROM CARRIERA AS C1 JOIN POSSESSIONE AS P1 ON C1.IDCARRIERA=P1.IDCARRIERA
WHERE P1.CF=CODICEF;
--Prelevo l'ultimo stipendio
SELECT C1.StipendioCorrente, C1.PrimaAssunzione INTO MyStip,PrimaA
FROM (CARRIERA AS C1 JOIN POSSESSIONE AS P1 ON C1.IDCARRIERA=P1.IDCARRIERA)
WHERE P1.CF=CODICEF AND C1.IDCARRIERA=MaxPromozione;
--Inserisco una nuova carriera
ora=now();
INSERT INTO
CARRIERA(PrimaAssunzione,DataPromozione,AumentoStipendio,RuoloPrecedente,StipendioPrecedente,
StipendioCorrente)
(SELECT PrimaA, ora, valore, D.Ruolo, MyStip, (MyStip*valore)+MyStip
FROM DIPENDENTE AS D
WHERE D.CF=CODICEF);
--Prendo l'ultima carriera inserita
SELECT MAX(C1.IDCARRIERA) INTO MaximumC
FROM CARRIERA AS C1;
--Inserisco una nuova tupla in possessione
INSERT INTO POSSESSIONE(IDCarriera,CF)
VALUES(MaximumC,CODICEF);
return true;
END;
$$
language plpgsql;
```

```
--Questa procedura permette di avere un feedback visivo riguardo alla promozione del salario
di un dipendente
CREATE OR REPLACE PROCEDURE Chiama_Aumento_Stipendio(CODICEF IN Dipendente.CF%TYPE, valore IN
float) AS $$
DECLARE
aumentato BOOLEAN;
BEGIN
RAISE NOTICE 'Aumento lo stipendio del dipendente % in percentuale %', CODICEF, valore;
SELECT Aumento_Stipendio(CODICEF,valore) INTO aumentato;
IF(aumentato) THEN
RAISE NOTICE 'Lo stipendio è stato aumentato.';
ELSE RAISE NOTICE 'Lo stipendio non è stato aumentato.';
END IF;
END;
$$
language plpgsql;
```

Esempio di cosa accade quando aumento lo stipendio di un dipendente:

```
1  call chiama_aumento_stipendio('F012345678901234',0.1);
```

Data output | Messages | Notifications

```
NOTIFICA:  Aumento lo stipendio del dipendente F012345678901234 in percentuale 0.1
NOTIFICA:  Lo stipendio è stato aumentato.
CALL
```

52	746	2016-01-01 00:00:00	2023-01-16 15:49:46.843691	Dirigente	0.1	1996.5	1815

```
--Questa funzione serve per verificare se un dipendente è idoneo al passaggio di ruolo
CREATE OR REPLACE PROCEDURE Check_For_Promotion(CODICEF IN Dipendente.CF%TYPE) AS $$
DECLARE
passaggio BOOLEAN;
BEGIN
RAISE NOTICE 'Controllo se il dipendente % è idoneo al passaggio di ruolo', CODICEF;
--Utilizzo la funzione Anzianita
SELECT Anzianita(CODICEF) INTO passaggio;
IF(passaggio)THEN
RAISE NOTICE 'Passaggio di ruolo effettuato.';
ELSE RAISE NOTICE 'Passaggio di ruolo non effettuato.';
END IF;
END;
$$
language plpgsql;
-------------------------------------
--Questa funzione verifica se tutti i dipendenti di una certa azienda sono idonei al passaggio
di ruolo
CREATE OR REPLACE PROCEDURE Check_For_Global_Promotion(nome IN Azienda.NOME%TYPE, via IN
Azienda.VIA%TYPE) AS $$
DECLARE
--Scrivo il comando per prelevare ogni codice fiscale di dipendenti di una certa azienda
comandoSQL VARCHAR(1000):='SELECT A.CF
                           FROM ASSUNZIONE AS A
                           WHERE A.VIA=$1 AND A.NOME=$2';
cursore REFCURSOR;
CODICEF Dipendente.CF%TYPE;
BEGIN
--apro il cursore utilizzando i parametri di ingresso
OPEN cursore FOR EXECUTE comandoSQL using via, nome;
LOOP
--prelevo il codice fiscale fin quando finiscono i dipendenti
FETCH cursore INTO CODICEF;
EXIT WHEN NOT FOUND;
--chiamo la funzione Check_For_Promotion
call Check_For_Promotion(CODICEF);
END LOOP;
CLOSE cursore;
END;
$$
language plpgsql;
```

Esempio di cosa accade quando verifico se i dipendenti di un'azienda sono idonei o meno alla promozione (passaggio di ruolo):

```
1  call check_for_global_promotion('Walmart','Ave Hostos');
```

Data output | Messages | Notifications

```
NOTIFICA:  Controllo se il dipendente F123456789012345 è idoneo al passaggio di ruolo
NOTIFICA:  Il dipendente F123456789012345 ha lavorato per 7 anni
NOTIFICA:  Aumento lo stipendio del dipendente F123456789012345 in percentuale 0.1
NOTIFICA:  Lo stipendio è stato aumentato.
NOTIFICA:  Aumento lo stipendio del dipendente F123456789012345 in percentuale 0.2
NOTIFICA:  Lo stipendio è stato aumentato.
NOTIFICA:  Passaggio di ruolo effettuato.
NOTIFICA:  Controllo se il dipendente F234567890123456 è idoneo al passaggio di ruolo
NOTIFICA:  Il dipendente F234567890123456 ha lavorato per 7 anni
NOTIFICA:  Passaggio di ruolo non effettuato.
NOTIFICA:  Controllo se il dipendente F345678901234567 è idoneo al passaggio di ruolo
NOTIFICA:  Il dipendente F345678901234567 ha lavorato per 7 anni
NOTIFICA:  Aumento lo stipendio del dipendente F345678901234567 in percentuale 0.1
NOTIFICA:  Lo stipendio è stato aumentato.
```

4.2.7 Consistenza data di prima assunzione

```
--Consistenza Prima Assunzione: questa trigger function verifica che
-- la data di firma di un contratto coincida con la data di prima assunzione di una carriera
CREATE OR REPLACE FUNCTION Consistenza_Prima_Assunzione() RETURNS TRIGGER AS $$
DECLARE
MyFirma Contratto.DataFirma%TYPE;
MyAssunzione Carriera.PrimaAssunzione%TYPE;
BEGIN
--Prelevo la data del primo contratto firmato dal dipendente
SELECT MIN(C1.DataFirma) INTO MyFirma
FROM CONTRATTO AS C1 JOIN DIPENDENTE AS D ON C1.CF=D.CF;
--Prelevo la prima assunzione della carriera che sto inserendo
SELECT C3.PrimaAssunzione INTO MyAssunzione
FROM CARRIERA AS C3
WHERE C3.IDCARRIERA=NEW.IDCARRIERA;
--Se le due date non coincidono non va bene
IF MyFirma!=MyAssunzione THEN
RAISE EXCEPTION 'La data di prima assunzione non coincide con la data di firma del primo
contratto.';
END IF;
return new;
END;
$$
language plpgsql;

CREATE OR REPLACE TRIGGER Trigger_Consistenza_Prima_Assunzione BEFORE INSERT ON POSSESSIONE
FOR EACH ROW
EXECUTE PROCEDURE Consistenza_Prima_Assunzione();
```

Esempio di cosa accade quando provo a collegare una carriera con data di prima assunzione diversa dalla data di firma del primo contratto di un dipendente:

```
1  INSERT INTO POSSESSIONE(IDCarriera,CF)
2  VALUES(797,'F345678901234567');
```

Data output | Messages | Notifications

```
ERROR: ERRORE:  La data di prima assunzione non coincide con la data di firma del primo contratto.
CONTEXT:  funzione PL/pgSQL consistenza_prima_assunzione() riga 15 a RAISE
```

4.2.8 Responsabilità progetti

```
--Responsabilità progetti: questa trigger function verifica che
-- all'inserimento di una tupla responsabilità il dipendente sia Dirigente
CREATE OR REPLACE FUNCTION Chiamata_Trigger_Reponsabilita_Progetti() RETURNS TRIGGER AS $$
DECLARE
MyRuolo Dipendente.Ruolo%TYPE;
BEGIN
--Prelevo il ruolo del dipendente che sto inserendo in responsabilità
SELECT D.Ruolo INTO MyRuolo
FROM DIPENDENTE AS D
WHERE New.CF=D.CF;
--Se il ruolo non è Dirigente non va bene
IF(MyRuolo!='Dirigente') THEN
RAISE EXCEPTION 'Il responsabile non è un Dirigente.';
END IF;
return new;
END;
$$
language plpgsql;

CREATE OR REPLACE TRIGGER Tigger_Responsabilita_Progetti BEFORE INSERT ON RESPONSABILITA
FOR EACH ROW
EXECUTE PROCEDURE Chiamata_Trigger_Reponsabilita_Progetti();
```

Esempio di cosa accade se provo a responsabilizzare un dipendente non Dirigente:

```
1  INSERT INTO RESPONSABILITA(CF,CUP)
2  VALUES('F345678901234567','CUP_1');
```

Data output | Messages | Notifications

```
ERROR: ERRORE:  Il responsabile non è un Dirigente.
CONTEXT:  funzione PL/pgSQL chiamata_trigger_reponsabilita_progetti() riga 11 a RAISE
```

4.2.9 Gestione Laboratori

```
--Gestione laboratorio: questa trigger function verifica che all'inserimento di una tupla
gestione il dipendente sia Senior
CREATE OR REPLACE FUNCTION Chiamata_Trigger_Gestione_Laboratorio() RETURNS TRIGGER AS $$
DECLARE
MyRuolo Dipendente.Ruolo%TYPE;
BEGIN
--Prelevo il ruolo del dipendente che sto inserendo in gestione
SELECT D.Ruolo INTO MyRuolo
FROM DIPENDENTE AS D
WHERE New.CF=D.CF;
--Se il ruolo non è Senior non va bene
IF(MyRuolo!='Senior') THEN
RAISE EXCEPTION 'Chi gestisce il laboratrio non è un dipendente Senior.';
END IF;
return new;
END;
$$
language plpgsql;


CREATE OR REPLACE TRIGGER Trigger_Gestione_Laboratorio BEFORE INSERT ON GESTIONE
FOR EACH ROW
EXECUTE PROCEDURE Chiamata_Trigger_Gestione_Laboratorio();
```

Esempio di cosa accade se provo a far gestire un laboratorio da un dipendente non Senior:

```
1  INSERT INTO GESTIONE(CF,Topic,Edificio,Stanza)
2  VALUES('ABCDEFGHIJKLMNOP','Informatica', 'Ingegneria', 'A103');
```

Data output | Messages | Notifications

```
ERROR: ERRORE:  Chi gestisce il laboratrio non è un dipendente Senior.
CONTEXT:  funzione PL/pgSQL chiamata_trigger_gestione_laboratorio() riga 11 a RAISE
```

4.2.10 Referenza Progetti

```
--Referenza progetti: questa trigger function verifica che all'inserimento di una tupla
referenza il dipendente sia Senior
CREATE OR REPLACE FUNCTION Chiamata_Trigger_Referenza_Progetti() RETURNS TRIGGER AS $$
DECLARE
MyRuolo Dipendente.Ruolo%TYPE;
BEGIN
--Prelevo il ruolo del dipendente che sto inserendo in referenza
SELECT D.Ruolo INTO MyRuolo
FROM DIPENDENTE AS D
WHERE New.CF=D.CF;
--Se in ruolo non è Senio non va bene
IF(MyRuolo!='Senior') THEN
RAISE EXCEPTION 'Il referente del progetto non è un dipendente Senior.';
END IF;
return new;
END;
$$
language plpgsql;


CREATE OR REPLACE TRIGGER Trigger_Referenza_Progetti BEFORE INSERT ON REFERENZA
FOR EACH ROW
EXECUTE PROCEDURE Chiamata_Trigger_Referenza_Progetti();
```

Esempio di cosa accade se provo a far referenziare un progetto da un dipendente non Senior:

```
1  INSERT INTO REFERENZA(CF,CUP)
2  VALUES('ABCDEFGHIJKLMNOP','CUP_1');
```

Data output | Messages | Notifications

```
ERROR: ERRORE:  Il referente del progetto non è un dipendente Senior.
CONTEXT:  funzione PL/pgSQL chiamata_trigger_referenza_progetti() riga 11 a RAISE
```