UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Informatica

Documento della base di dati

Sistema di gestione di personale e progetti all'interno di un'azienda

Anno Accademico 2022/2023

Autori
Tortora Salvatore, Terrecuso Francesco
Relative matricole: N86004033, N86004191

Indice

1 Modello Concettuale

1.1 Analisi dei Requisiti

In questa sezione si analizza la richiesta del cliente, con lo scopo di definire le funzionalità che la base di dati deve soddisfare. Individueremo le entità e le associazioni del mini-word, per rappresentare in modo esaustivo e chiaro il dominio del problema.

> *Si sviluppi un sistema informativo, composto da una base di dati relazionale e da un applicativo Java dotato di GUI (Swing o JavaFX), per la gestione del personale di un'azienda. L'azienda possiede un certo numero di impiegati, raggruppabili in 4 categorie:*
>
> - *Dipendente Junior : colui che lavora da meno di tre anni nell'Azienda.*
> - *Dipendente Middle : colui che lavora da più di tre anni nell'Azienda. ma meno di sette.*
> - *Dipendente Senior : colui che lavora da più di sette anni nell'Azienda.*
> - *Dirigenti: la classe dirigente non ha obblighi temporali di servizio. Chiunque può diventare dirigente, se mostra di averne le capacità.*
>
> *I passaggi di ruolo avvengono per anzianità di servizio.*

L'incipit della richiesta è chiara : fornire una base di Dati per la Gestione del personale di un Azienda. Le Entità principali da rappresentare nel nostro mini-word sono le seguenti : L'Impiegato e le sue specializzazioni, in particolare, le entità Junior, Middle e Senior. La prima nota che risalta dalla richiesta del cliente è quella di dividere la tipologia di Impiegato per anni di appartenenza all'azienda specificando che gli scatti possibili di carriera sono svincolati dalle competenze effettive dell'impiegato e sono sanciti solamente dagli anni di lavoro. L'entità Dirigente fa eccezione da tali specializzazioni in quanto il suo ruolo non dipende dal tempo di appartenenza all'Azienda bensì dalle capacità ben specifiche.

> *È necessario tracciare tutti gli scatti di carriera per ogni dipendente.*

La richiesta è modellabile con una classe Storico in cui salvare ogni scatto di carriera per ogni impiegato, la data e il passaggio di ruolo; Oltre alla rappresentazione degli scatti di carriera per le specializzazioni junior, middle e Senior, bisogna considerare anche gli scatti Dirigenziali e, siccome la richiesta non è specifica a riguardo, consideriamo che un Impiegato può avere più scatti Dirigenziali.

Nell'azienda vengono gestiti laboratori e progetti. Un laboratorio ha una particolare topic di cui si occupa, un certo numero di afferenti ed un responsabile scientifico che è un dipendente senior.

Abbiamo la presentazione di due nuove Entità del mini-word : Laboratorio e Progetto. L'attenzione è focalizzata sull'Entità Laboratorio, ad esso afferiscono gli Impiegati, ciò suggerisce l'associazione fra le due entità e si suppone che un impiegato può afferire a più laboratori; E' specificato anche che oltre gli afferenti, un laboratorio ha un responsabile scientifico il quale è Senior. Siccome la richiesta non è specifica a riguardo, si considera il responsabile scientifico univoco per ogni laboratorio, in quanto è specializzato nel topic di quest'utlimo.

Un progetto è identificato da un CUP (codice unico progetto) e da un nome (unico nel sistema). Ogni progetto ha un referente scientifico, il quale deve essere un dipendente senior dell'ente, ed un responsabile che è uno dei dirigenti.

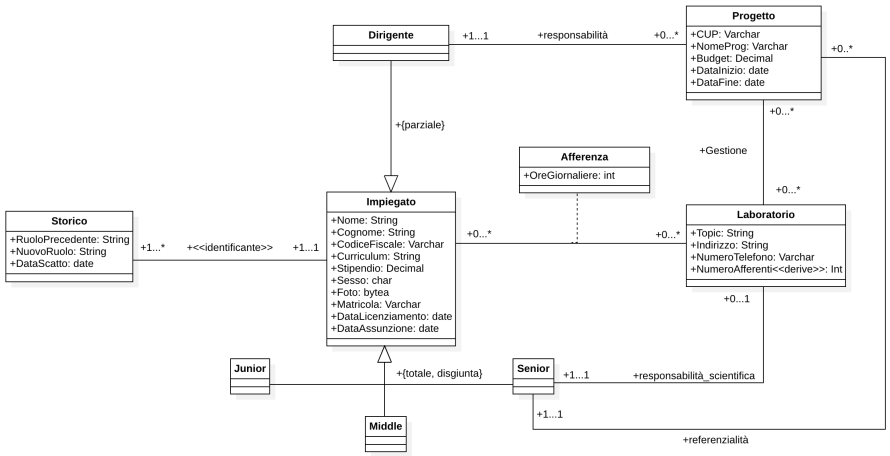
Il focus è spostato sulla nuova entità Progetto univocamente determinata da un attributo CUP e da un nome unico nel sistema. Da questa sezione riusciamo a capire che ogni progetto ha referente scientifico (Senior), e un responsabile (dirigente). Siccome la richiesta è vaga e non specifica, si suppone che un referente scientifico può essere associato a più progetti, in equal modo un Dirigente può essere associato a più progetti. Ciò suggerisce ovviamente le associazioni fra queste entità.

Al massimo 3 laboratori possono lavorare ad un progetto.

Questo punto è essenziale nel comprendere la relazione che esiste fra le entità Progetto e Laboratorio, specificandone i vincoli di gestione. Tuttavia la richiesta rimane vaga, motivo per il quale si consideri che un laboratorio può essere gestito da più progetti, mentre un progetto può gestire al più tre laboratori.

1.2 Diagramma (UML)

In seguito alle considerazioni espresse nella sezione precedente, si è prodotto il seguente schema concettuale espresso mediante diagramma UML:



2 Ristrutturazione del Modello Concettuale

Prima di poter passare allo schema logico è necessario ristrutturare il diagramma delle classi per semplificare la traduzione in schema logico, ottimizzare il progetto, eliminare le generalizzazioni, eliminare gli attributi multivalore, eliminare gli attributi strutturati, accorpare o partizionare le entità figlie e scegliere gli identifiatori delle nostre entità .

2.1 Analisi delle Ridondanze

Una ridondanza è un dato che è già presente nella base di dati o può essere derivato da altri dati.

Nel modello concettuale originale non vi sono ridondanze; L'unico attributo derivato è 'Numero Afferenti' all'interno dell'entità Laboratorio. Questo attributo serve ad identificare quanti sono gli afferenti di un dato laboratorio ed è semplice ricavarlo attraverso il conteggio delle occorrenze fra il Laboratorio e gli Impiegati che vi afferiscono; Siccome i Laboratori sono gestiti da Progetti, per conoscere quanti sono gli Impiegati che lavorano ad un progetto bisognerebbe,senza questo attributo, passare per il conteggio delle varie occorrenze dei laboratori che lavorano al progetto, ciò viene semplificato inserendo l'attributo derivato in questione.

2.2 Analisi delle Generalizzazioni

La specializzazione è il processo di determinazione di sottoclassi per una data entità. La generalizzazione è il suo concetto complementare. Nel modello concettuale vi sono due specializzazioni diverse per la classe Impiegato.
1.La specializzazione (totale, disgiunta) delle sotto-classi Junior, Middle e Senior che risolviamo accorpando queste classi figlie in quella padre Impiegato, inserendo un nuovo attributo 'tipo impiegato'.
2.La specializzazione (parziale) della sotto-classe Dirigente che modelliamo accorpandola all'entità genitore, esprimendola invece come una singola flag booleana ' dirigente ';

2.3 Eliminazione degli attributi Multivalore o Composti

Un attributo è multivalore se può essere associato ad un numero variabile di valori dello stesso dominio. Un attributo è composto se può essere suddiviso in sottoparti ognuna dotata di dominio.

Nel modello concettuale non sono presenti attributi multivalore o composti.

2.4 Analisi di Entità e Associazioni

Non si è ritenuto necessario scomporre o accorpare entità.

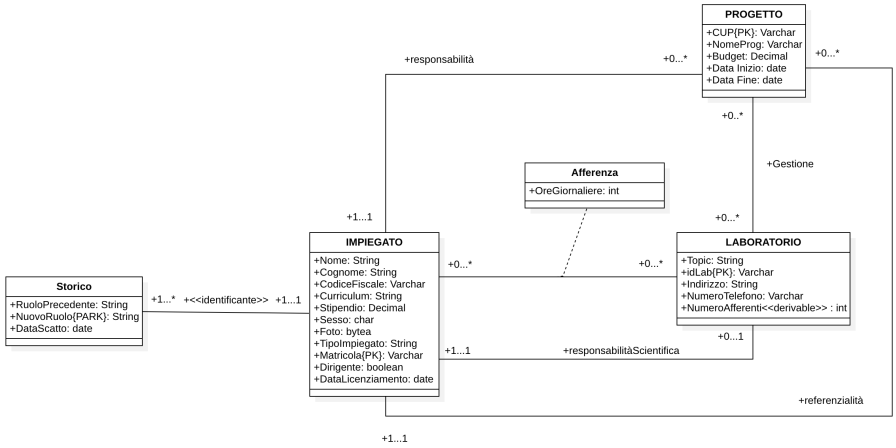
2.5 Scelta degli Identificatori Primari

Un identificatore o chiave minimale è un insieme di attributi che individuano univocamente un'entità. È possibile che un'entità sia dotata di molteplici identificatori, fra i quali ne sceglieremo uno principale che agirà da chiave primaria.

- **Impiegato**: fra le chiavi candidate Codice Fiscale e Matricola è stato scelta come chiave primaria Matricola in quanto più breve e indica naturalmente l'identificativo di ogni Impiegato.
- **Laboratorio**: si è deciso d'introdurre l'identificativo IdLab in quanto dai requisiti non si evince un attributo che identifica univocamente un laboratorio.
- **Progetto**: dalla traccia sono specificate due chiavi candidate precise : il CUP (codice univoco progetto) e il nome del progetto in quanto unico nel sistema. Si è preferito scegliere come chiave primaria il CUP.
- **Storico**: esso è un entità debole e, in quanto tale, non ha un attributo che identifica gli scatti di carriera poiché quest'ultimi non hanno motivo di esistere se non vi è un Impiegato a compierli; Comunque si decide come chiave parziale il nuovo ruolo e la data scatto.

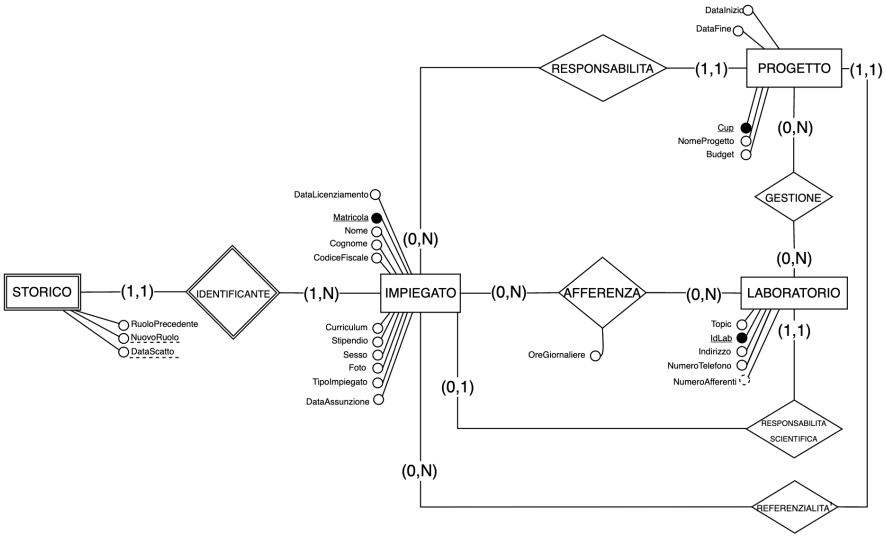
2.6 Diagramma Ristrutturato (UML)

Dopo aver ristrutturato il Class Diagram come descritto precedentemente, possiamo produrre il seguente schema concettuale ristrutturato espresso mediante diagramma UML:



2.7 Diagramma Ristrutturato (ER)

Ulteriore notazione per poter esprimere lo schema ristrutturato è il seguente l'ER :



3 Dizionari

3.1 Dizionario delle Entità

Entità	Descrizione	Attributi
Impiegato	Entità che fa parte del personale dell'Azienda	**Matricola**(Stringa): Identificante di un impiegato. **Nome**(Stringa): Nome di un impiegato. **Cognome**(Stringa): Cognome di un impiegato. **Codice Fiscale**(Stringa): codice fiscale di un impiegato. **Curriculum**(Stringa): Descrive le competenze di un Impiegato. **Stipendio**(Stringa): Descrive la somma complessiva dello stipendio. **Sesso**(Carattere): Specifica il sesso di un Impiegato. **Foto**(Stringa binaria): rappresenta la foto dell'impiegato. **tipo_impiegato**(stringa):rappresenta il ruolo attuale dell'impiegato. **Dirigente**(booleano):indica se attualmente l'impiegato è dirigente oppure no. **Data assunzione**(data): Descrive la data di assunzione di un Impiegato. **Data licenziamento**(Data): data di licenziamento dell'impiegato.
Storico	Entità Debole in cui si salvano tutti gli scatti di carriera di un impiegato	**Ruolo precedente**(Stringa): ruolo precedente allo scatto. **Nuovo ruolo**(Stringa): il ruolo di scatto. **Data scatto**(data): Data dello scatto di ruolo.

Entità	Descrizione	Attributi
Afferenza	Entità ad associazione tra Laboratorio e Impiegato.	**Ore giornaliere**(intero): indica le ore lavorative giornaliere su un laboratorio.
Laboratorio	Luogo in cui lavorano gli Impiegati	**Topic**(Stringa): Indica la materia trattata nel laboratorio. **Indirizzo**(Stringa): Indica la locazione del laboratorio. **Numero Telefononico**(intero): Descrive il riferimento telefonico. **Numero Afferenti**(Intero): Attributo derivato che descrive quanti afferenti ha un laboratorio.
Progetto	Ideazione e progettazione di un'Idea lavorativa	**CUP**(stringa): Codice Univoco del Progetto. **Nome Progetto**(stringa): Il nome del progetto unico nel sistema. **Budget**(Decimale):Capitale investito nel progetto. **Data Inizio**(data): data di avvio del progetto. **Data Fine**(data): data di fine progetto.

3.2 Dizionario delle Associazioni

Associazione	Tipologia	Descrizione
Afferenza	Molti-a-Molti	Associa ad un impiegato all'insieme di laboratori a cui lavora.
Responsabilità	Uno-a-Molti	Associa un Dirigente ai suoi progetti.
Responsabilità scientifica	Uno-a-Uno	Associa un laboratorio al proprio responsabile scientifico.
Referenzialità	Uno-a-Molti	Associa un referente ai suoi progetti.
Gestione	Molti-a-Molti	Un laboratorio può essere gestito da più progetti, un progetto gestisce al massimo tre laboratori.
Identificante	Uno-a-Molti	Associa un Impiegato ai suoi scatti di carriera.

3.3 Dizionario dei Vincoli

Vincolo	Tipologia	Descrizione
Di Gestione	interrelazione	Un progetto attivo può gestire al più tre laboratori.
Di Referenzialità	interrelazionale	Il referente di un progetto è un dipendente Senior
Di Responsabilità	interrelazionale	Il responsabile di un progetto è un dirigente dell'Azienda.
Di Responsabilità Scientifico	interrelazionale	Il responsabile scientifico di un laboratorio è un dipendente Senior
Di Unicità del Nome Progetto	intrarelazionale	Il nome di un progetto è UNICO nel Sistema.
Su ore lavorative	interrelazionale	Un impiegato non può lavorare per più di otto ore al giorno.

4 Modello Logico

4.1 Traduzione di Entità e Associazioni

Finito il processo di ristrutturazione, possiamo procedere col *mapping* di entità e associazioni. Per ogni entità del modello ristrutturato, definiremo una relazione equivalente con gli stessi attributi. Il processo di traduzione per le associazioni è invece più complesso e richiede un'analisi più approfondita.

- **Identificante**: è un tipo di associazione *uno-a-molti totale*, motivo per il quale si è deciso di mappare tale associazione usando nell'entità Storico la chiave esterna matricola, in quanto un Impiegato ha sicuramente almeno uno scatto di carriera ma soprattutto poiché lo Storico è un'entità debole e ha bisogno della chiave primaria di un impiegato per identificare ogni suo scatto.

- **Afferenza,Gestione**: sono associazioni *molti-a-molti*, motivo per il quale vengono espresse tramite relazioni indipendenti.

- **Responsabilità, Referenzialità**: essendo queste associazioni *uno-a-molti parziali* rispetto al progetto il quale ha un referente ed un responsabile, si è deciso di mappare tali associazioni attraverso l'introduzione delle due chiavi esterne all'interno dell'Entità Progetto.

- **Responsabilità scientifica**: tale associazione *uno-a-uno parziale*, si sceglie di mappare anch'essa attraverso l'introduzione della chiave esterna del responsabile all'interno della classe Laboratorio.

4.2 Relazioni

Legenda: Chiave Primaria, Chiave Esterna↑, Attributo Derivato

- **IMPIEGATO**(Matricola, Nome, Cognome, Codice fiscale, Curriculum, Stipendio, Foto, Tipo impiegato, Dirigente, Data assunzione, Data licenziamento.)

- **LABORATORIO**(IdLab, Topic, Indirizzo, Numero telefono, Numero afferenti,Responsabile Scientifico↑)
  - Responsabile Scientifico → IMPIEGATO.Matricola

- **PROGETTO**(Cup, Referente↑, Responsabile↑,Nome progetto,Budget, Data Inizio, Data Fine)
  - Referente → IMPIEGATO.Matricola
  - Responsabile → IMPIEGATO.Matricola

- **STORICO**(Matricola↑, Data scatto, Nuovo ruolo,Ruolo precedente)
  - Matricola → IMPIEGATO.Matricola

- **AFFERENZA**(Matricola↑, IdLab↑,Ore giornaliere)
  - Matricola → IMPIEGATO.Matricola
  - IdLab → LABORATORIO.Idlab

- **GESTIONE**(matricola↑, Idlab↑)
  - Matricola → IMPIEGATO.Matricola
  - IdLab → LABORATORIO.Idlab

5 Modello Fisico

5.1 Domini

```
1 CREATE DOMAIN DOMINIO_SESSO AS CHAR
2     CHECK(VALUE IN('M','F','N'))
```

Il dominio DOMINIO_SESSO indica il corretto dominio che può assumere l'attributo sesso in IMPIEGATO.

```
1 CREATE DOMAIN DOMINIO_MATRICOLA AS
2     VARCHAR(8) CHECK(VALUE LIKE('MAT-%'));
```

DOMINIO_MATRICOLA indica la forma corretta che l'attributo Matricola può assumere, in particolare il numero di matricola sarà anticipato dal prefisso 'MAT-'.

```
1 CREATE DOMAIN DOMINIO_IMPIEGATO AS VARCHAR
2     CHECK(VALUE IN('junior','middle','senior'));
```

Il Dominio sopra riportato è specifico per l'attributo *Tipo impiegato*.

```
1 CREATE DOMAIN DOMINIO_SCATTO AS
2     VARCHAR CHECK(VALUE IN('junior','middle','senior',
3                            'dirigente','NonDirigente'));
```

Questo Dominio è importante per descrivere il tipo di scatto che un impiegato può fare, in particolare rispetto al DOMINIO_IMPIEGATO, vi si aggiungono due nuovi tipi : dirigente e NonDirigente, in modo tale da salvare all'interno dello STORICO, tutti gli scatti di carriera anche quelli dirigenziali i quali, come descritto nella sezione Analisi dei Requisiti , possono essere molteplici motivo per il quale non devo solamente tener traccia della data in cui inizia ad essere Dirigente, ma anche della data in cui smette di esserlo.

5.2 Tabelle

TABELLA IMPIEGATO :

```
1  CREATE TABLE IF NOT EXISTS IMPIEGATO(
2          matricola DOMINIO_MATRICOLA,
3          nome VARCHAR NOT NULL,
4          cognome VARCHAR NOT NULL,
5          codice_fiscale CHAR(16) NOT NULL UNIQUE,
6          curriculum VARCHAR,
7          stipendio DECIMAL(12,2) NOT NULL,
8          sesso DOMINIO_SESSO NOT NULL,
9          foto BYTEA,
10         tipo_impiegato DOMINIO_IMPIEGATO NOT NULL DEFAULT 'junior',
11         dirigente BOOLEAN NOT NULL DEFAULT FALSE,
12         data_assunzione DATE NOT NULL
13         data_licenziamento DATE DEFAULT NULL,
14
15         CONSTRAINT data_corretta CHECK(data_assunzione < data_licenziamento),
16         CONSTRAINT impiegato_pk PRIMARY KEY(matricola),
17         CONSTRAINT stipendio_corretto CHECK(stipendio > 0)
18 );
```

nota: *data_corretta* è un vincolo che definisce la seguente regola : la data assunzione dev'essere precedente rispetto alla data di licenziamento.

TABELLA LABORATORIO:

```
1  CREATE TABLE IF NOT EXISTS LABORATORIO(
2          id_lab VARCHAR,
3          topic VARCHAR NOT NULL,
4          indirizzo VARCHAR NOT NULL,
5          numero_telefono VARCHAR(12),
6          numero_afferenti INTEGER DEFAULT 1,
7          r_scientifico DOMINIO_MATRICOLA NOT NULL UNIQUE,
8
9          CONSTRAINT laboratorio_pk PRIMARY KEY(id_lab),
10         CONSTRAINT r_scientifico_fk FOREIGN KEY(r_scientifico)
11         REFERENCES IMPIEGATO(matricola)
12                 ON UPDATE CASCADE
13 );
```

nota: *r_scientifico* identifica il responsabile scientifico di un laboratorio e, siccome si è considerato unico per ogni laboratorio, allora vi è stato aggiunto la clausola di UNIQUE.

TABELLA PROGETTO :

```
CREATE TABLE IF NOT EXISTS PROGETTO(
        cup VARCHAR,
        nome_progetto VARCHAR UNIQUE NOT NULL,
        budget DECIMAL(12,2) NOT NULL,
        data_inizio DATE NOT NULL,
        data_fine DATE DEFAULT NULL,
        responsabile DOMINIO_MATRICOLA NOT NULL,
        referente DOMINIO_MATRICOLA NOT NULL,

        CONSTRAINT progetto_pk PRIMARY KEY(cup),
        CONSTRAINT budget_corretto CHECK(budget > 0),
        CONSTRAINT data_corretta CHECK(data_fine > data_inizio),
          CONSTRAINT responsabile_fk FOREIGN KEY(responsabile)
        REFERENCES IMPIEGATO(matricola)
                ON UPDATE CASCADE,
          CONSTRAINT referente_fk FOREIGN KEY(referente)
        REFERENCES IMPIEGATO(matricola)
                ON UPDATE CASCADE
);
```

nota: con referente , s'intende il referente scientifico del progetto.

TABELLA AFFERENZA :

```
CREATE TABLE IF NOT EXISTS AFFERENZA(
        ore_giornaliere INTEGER NOT NULL,
        matricola DOMINIO_MATRICOLA NOT NULL,
        id_lab VARCHAR NOT NULL,

        CONSTRAINT afferenza_pk PRIMARY KEY(matricola,id_lab),
        CONSTRAINT impiegato_fk FOREIGN KEY(matricola)
        REFERENCES IMPIEGATO(matricola)
                ON DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT laboratorio_fk FOREIGN KEY(id_lab)
        REFERENCES LABORATORIO(id_lab)
                ON UPDATE CASCADE ON DELETE CASCADE

);
```

nota: l'attributo *ore_giornaliere,* indica le ore lavorative di un impiegato su un singolo laboratorio.

TABELLA STORICO :

```
CREATE TABLE IF NOT EXISTS STORICO(
        ruolo_prec DOMINIO_SCATTO,
        nuovo_ruolo DOMINIO_SCATTO NOT NULL,
        data_scatto DATE NOT NULL,
        matricola DOMINIO_MATRICOLA,

        CONSTRAINT storico_pk PRIMARY KEY(nuovo_ruolo,matricola,data_scatto),
        CONSTRAINT matricola_fk FOREIGN KEY(matricola)
        REFERENCES IMPIEGATO(matricola)
                ON UPDATE CASCADE ON DELETE CASCADE,
        CONSTRAINT check_ruolo CHECK(
    ((ruolo_prec is NULL) AND (nuovo_ruolo = 'junior')) OR
    ((ruolo_prec = 'junior') AND (nuovo_ruolo = 'middle')) OR
    ((ruolo_prec = 'middle') AND (nuovo_ruolo = 'senior')) OR
    (ruolo_prec = 'NonDirigente') AND (nuovo_ruolo = 'dirigente') OR
    ((ruolo_prec = 'dirigente') AND (nuovo_ruolo = 'NonDirigente'))) );
```

nota: il vincolo *check_ruolo* , descrive le sole ed uniche possibilità di combinazione tra gli scatti di carriera, definendo in questo modo le sequenze possibili di promozioni.

TABELLA GESTIONE :

```
CREATE TABLE IF NOT EXISTS GESTIONE(
        cup VARCHAR NOT NULL,
        id_lab VARCHAR NOT NULL,

        CONSTRAINT gestione_pk PRIMARY KEY(cup, id_lab),
        CONSTRAINT progetto_fk FOREIGN KEY(cup)
        REFERENCES PROGETTO(cup)
                ON UPDATE CASCADE ON DELETE CASCADE ,
        CONSTRAINT laboratorio_fk FOREIGN KEY(id_lab)
        REFERENCES LABORATORIO(id_lab)
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

5.3 Viste

In questa sezione vengono descritte l'insieme di *view* che vengono messe a disposizione a chi nell'azienda ha il compito di gestire il personale, i progetti e i laboratori.

```
1 CREATE OR REPLACE VIEW Impiegati_attuali AS(
2 SELECT*
3 FROM Impiegato
4 where data_licenziamento IS NULL or
      data_licenziamento > CURRENT_DATE);
```

La vista *Impiegati_attuali* descrive semplicemente la lista di impiegati che non sono stati licenziati, dove la data di licenziamento è messa a NULL oppure non si è ancora raggiunto la fine del contratto (si pensi nel caso in cui l'impiegato ha un particolare tipo di contratto determinato in cui sin dall'inserimento nell'azienda conosce la data di fine del proprio percorso lavorativo[...]).

```
1 CREATE OR REPLACE VIEW Dirigenti_Attuali AS (
2     SELECT *
3     FROM Impiegati_attuali
4     WHERE dirigente is true );
```

Dalla vista sugli Impiegati Attuali, ne deduciamo i dirigenti attuali.

```
1 CREATE OR REPLACE VIEW Referenti_attuali AS(
2     SELECT *
3     FROM Impiegati_attuali as i
4     WHERE i.matricola IN (SELECT referente FROM
          Progetti_attuali));
```

La vista rappresenta i referenti attuali dei progetti in corso.

```
1 CREATE OR REPLACE VIEW
    Responsabili_scientifici_attuali AS(
2     SELECT *
3     FROM Impiegati_attuali as i
4     WHERE i.matricola IN (SELECT r_scientifico FROM
          laboratorio));
```

La vista rappresenta i responsabili scientifici attuali dei laboratori.

```
1 CREATE OR REPLACE VIEW Afferenza_attuale AS(
2     SELECT a.matricola, a.id_lab
3     FROM afferenza as a NATURAL JOIN
          Impiegati_attuali );
```

La vista *Afferenza_attuale,* mostra un impiegato attuale rispetto ai laboratori in cui lavora.

```
1 CREATE OR REPLACE VIEW Progetti_attuali AS(
2     SELECT *
3     FROM progetto
4     WHERE data_fine > CURRENT_DATE OR data_fine is
          NULL );
```

La vista *Progetti_Attuali* descrive i progetti che sono attivi nell'azienda.

```
1 CREATE OR REPLACE VIEW Progetti_terminati AS(
2     SELECT *
3     FROM progetto
4     WHERE data_fine < CURRENT_DATE);
```

Di conseguenza è utile avere una vista di tutti i progetti terminati.

```
1 CREATE OR REPLACE VIEW Storico_view AS (
2     SELECT i.nome, i.cognome, i.matricola,
3         s1.data_scatto AS data_scatto_junior,
4         s2.data_scatto AS data_scatto_middle,
5         s3.data_scatto AS data_scatto_senior
6     FROM IMPIEGATO i
7     LEFT JOIN STORICO s1 ON i.matricola = s1.
          matricola
8     AND s1.nuovo_ruolo = 'junior'
9     LEFT JOIN STORICO s2 ON i.matricola = s2.
          matricola
10    AND s2.nuovo_ruolo = 'middle'
11    LEFT JOIN STORICO s3 ON i.matricola = s3.
          matricola
12    AND s3.nuovo_ruolo = 'senior'
13 );
```

La precedente vista *STORICO_VIEW*, mette a disposizione un'importante semplificazione della tabella STORICO, riuscendo a rappresentare in questo modo un impiegato con tutti i suoi scatti di carriera (se compiuti) in un'unica tupla.

```
1 CREATE OR REPLACE VIEW Gestione_Attuale AS (
2     SELECT g.cup,g.id_lab
3     FROM gestione as g NATURAL JOIN Progetti_attuali
          as p);
```

E' utile conoscere per un progetto non terminato, i laboratori che gestisce.

6 Triggers

In questa sezione verranno mostrati tutti i triggers implementati per ottenere un database **consistente** e che continua a funzionare nel tempo, assicurando di avere dei **dati concreti e attendibili**.

6.1 Triggers su Impiegato

- Il trigger **insert_storico** si occupa di controllare se all'inserimento di un impiegato la data_assunzione è coerente con l'attributo tipo_impiegato, nel caso affermativo calcola e inserisce tutti gli scatti di carriera fatti.

```
1  CREATE OR REPLACE TRIGGER insert_storico
2  AFTER INSERT ON impiegato
3  FOR EACH ROW
4  EXECUTE FUNCTION f_insert_storico();
5
6  CREATE OR REPLACE FUNCTION f_insert_storico() RETURNS TRIGGER AS
7  $$
8  BEGIN
9      IF (NEW.tipo_impiegato = 'junior') THEN
10         IF((NEW.data_assunzione + INTERVAL '3 years') >= CURRENT_DATE ) THEN
11             INSERT INTO storico VALUES (NULL, 'junior', NEW.data_assunzione,
                   NEW.matricola);
12         ELSE
13             RAISE EXCEPTION 'DATA DI ASSUNZIONE NON VALIDA PER UN DIPENDENTE
                   JUNIOR';
14         END IF;
15
16     ELSIF (NEW.tipo_impiegato = 'middle') THEN
17         IF ((NEW.data_assunzione + INTERVAL '3 years') <= CURRENT_DATE AND
18             NEW.data_assunzione + INTERVAL '7 years' >= CURRENT_DATE ) THEN
19                 INSERT INTO storico VALUES (NULL, 'junior', NEW.
                       data_assunzione, NEW.matricola);
20                 INSERT INTO storico VALUES ('junior', 'middle', NEW.
                       data_assunzione + INTERVAL '3 years', NEW.matricola);
21         ELSE
22             RAISE EXCEPTION 'DATA DI ASSUNZIONE NON VALIDA PER UN DIPENDENTE
                   MIDDLE';
23         END IF;
24
25     ELSIF (NEW.tipo_impiegato = 'senior') THEN
26         IF((NEW.data_assunzione + INTERVAL '7 years') <= CURRENT_DATE ) THEN
27             INSERT INTO storico VALUES (NULL, 'junior', NEW.data_assunzione,
                   NEW.matricola);
28             INSERT INTO storico VALUES ('junior', 'middle', NEW.
                   data_assunzione + INTERVAL '3 years', NEW.matricola);
29             INSERT INTO storico VALUES ('middle', 'senior', NEW.
                   data_assunzione + INTERVAL '7 years', NEW.matricola);
30         ELSE
31             -- Error message
32             RAISE EXCEPTION 'DATA DI ASSUNZIONE NON VALIDA PER UN DIPENDENTE
                   SENIOR';
33         END IF;
34     END IF;
35
36     IF(new.dirigente is true) THEN
37             INSERT INTO STORICO VALUES('NonDirigente','dirigente',
                   CURRENT_DATE, new.matricola);
38         END IF;
```

```
39     RETURN NEW;
40 END;
41 $$ LANGUAGE plpgsql;
```

- Il trigger **update_dirigente** si occupa di inserire il nuovo scatto di carriera all'interno dello storico; Nel caso in cui si declassa un dirigente si verifica se è responsabile di qualche progetto attivo, nel caso affermativo lancia il messaggio di errore.

```
1  CREATE OR REPLACE TRIGGER update_dirigente
2  AFTER UPDATE OF dirigente On impiegato
3  FOR EACH ROW
4  EXECUTE FUNCTION f_update_dirigente();
5
6  CREATE OR REPLACE FUNCTION f_update_dirigente() RETURNS trigger AS
7  $$
8  BEGIN
9          IF((OLD.dirigente = false ) AND NEW.dirigente = true)THEN
10                 INSERT INTO STORICO VALUES('NonDirigente','dirigente',
                       CURRENT_DATE, new.matricola);
11
12         ELSIF (NEW.dirigente = false AND OLD.dirigente = true)THEN
13                 IF EXISTS(SELECT* FROM PROGETTO WHERE responsabile = new.
                       matricola AND (data_fine is null or data_fine >
                       CURRENT_DATE) ) THEN
14
15                     RAISE EXCEPTION 'NON PUOI DECLASSARE UN DIRIGENTE SE
                            GESTISCE UN PROGETTO, BISOGNA PRIMA CAMBIARLO!
                           ';
16
17
18                 ELSE
19                         INSERT INTO STORICO VALUES('dirigente','NonDirigente
                               ', CURRENT_DATE, new.matricola);
20                 END IF;
21
22         END IF;
23
24         RETURN NEW;
25
26 END;
27 $$ language plpgsql;
```

- Il trigger **not_update_tipo_impiegato** scatta alla modifica degli attributi tipo_impiegato e data_assunzione ne blocca la modifica, in modo da rendere consistente la base di dati.

```
1  CREATE OR REPLACE TRIGGER  not_update_tipo_impiegato
2  AFTER UPDATE OF tipo_impiegato,data_assunzione ON impiegato
3  FOR EACH ROW
4  EXECUTE FUNCTION f_not_update_tipo_impiegato();
5
6  CREATE OR REPLACE FUNCTION f_not_update_tipo_impiegato() RETURNS TRIGGER AS
7  $$
8  BEGIN
9          RAISE EXCEPTION 'UNA VOLTA INSERITO UN IMPIEGATO NON PUOI MODIFICARE
               IL SUO TIPO O LA DATA DI ASSUNZIONE.';
10         RETURN NEW;
11 END;
12 $$ LANGUAGE plpgsql;
```

- Il trigger **eliminazione_impiegati_speciali** impedisce l'eliminazione di impiegati attuali che sono referenti o responsabili di un progetto/laboratorio, inviando un messaggio di errore.

```
1  CREATE OR REPLACE TRIGGER eliminazione_impiegati_speciali
2  BEFORE DELETE ON IMPIEGATO
3  FOR EACH ROW
4  EXECUTE function f_eliminazione_impiegati_speciali();
5
6  CREATE OR REPLACE FUNCTION f_eliminazione_impiegati_speciali() RETURNS
       TRIGGER AS
7  $$
8  BEGIN
9          IF EXISTS(select* from PROGETTI_ATTUALI where responsabile = NEW.
               matricola) THEN
10
11         RAISE EXCEPTION 'IMPOSSIBILE ELIMINARE IL RESPONSABILE DI UN
               PROGETTO ATTIVO, PRIMA BISOGNA SOSTITUIRLO!';
12         END IF;
13
14         IF EXISTS(select* from PROGETTI_ATTUALI where referente = NEW.
               matricola) THEN
15
16         RAISE EXCEPTION 'IMPOSSIBILE ELIMINARE IL REFERENTE DI UN PROGETTO
               ATTIVO, PRIMA BISOGNA SOSTITUIRLO!';
17         END IF;
18
19         IF EXISTS(select* from LABORATORIO where r_scientifico = NEW.
               matricola) THEN
20         RAISE EXCEPTION 'IMPOSSIBILE ELIMINARE IL RESPONSABILE SCIENTIFICO
               DI UN LABORATORIO, PRIMA BISOGNA SOSTITUIRLO!';
21         END IF;
22         RETURN OLD;
23 END;
24 $$ language plpgsql;
```

- Il seguente trigger **check_stipendio** controlla la gerarchia degli stipendi ricevuti per ogni impiegato, verificando che un Junior riceve meno di un Middle che a sua volta riceve meno di un Senior. Da tale vincolo di integrità sematica fa eccezione il Dirigente.

```
1  CREATE OR REPLACE TRIGGER check_stipendio
2  AFTER INSERT OR UPDATE OF stipendio ON Impiegato
3  FOR EACH ROW
4  execute function f_check_stipendio();
5
6  CREATE OR REPLACE FUNCTION f_check_stipendio() RETURNS TRIGGER AS
7  $$
8  BEGIN
9          IF(NEW.dirigente is true)then
10                 RETURN NEW;
11         END IF;
12
13         IF(new.tipo_impiegato = 'junior' AND EXISTS(select* from Impiegato
               as i where i.tipo_impiegato <> 'junior' and i.stipendio < new.
               stipendio )) then
14
15                 RAISE EXCEPTION 'UN IMPIEGATO JUNIOR NON PUO AVERE LO
                       STIPENDIO PIU ALTO DI UN MIDDLE';
16
17         ELSIF(new.tipo_impiegato = 'middle' AND EXISTS(select* from
               Impiegato as i where i.tipo_impiegato ='senior' and i.stipendio
                < new.stipendio)) then
18
19                 RAISE EXCEPTION 'UN IMPIEGATO MIDDLE NON PUO AVERE LO
                       STIPENDIO PIU ALTO DI UN SENIOR';
20         END IF;
21
22         RETURN NEW;
23 END;
24 $$ LANGUAGE plpgsql;
```

- Il seguente trigger **avviso_su_impiegati_licenziati** parte nel caso in cui licenzio un impiegato; Se esso è un responsabile di un progetto o di un laboratorio o, nel caso in cui sia un referente, manda il messaggio di warning avvisando l'utente di aggiornare gli eventuali errori.

```
1  CREATE OR REPLACE TRIGGER avviso_su_impiegati_licenziati
2  AFTER UPDATE OF data_licenziamento ON impiegato
3  FOR EACH ROW
4  EXECUTE FUNCTION f_avviso_su_impiegati_licenziati();
5
6  CREATE OR REPLACE FUNCTION f_avviso_su_impiegati_licenziati() RETURNS
       TRIGGER AS
7  $$
8  DECLARE
9          impiegato RECORD;
10 BEGIN
11         IF NEW.data_licenziamento IS NOT NULL THEN
12
13                 SELECT l.r_scientifico, l.id_lab INTO impiegato
14                 FROM laboratorio AS l
15                 WHERE NEW.matricola = l.r_scientifico
16                 AND l.r_scientifico NOT IN (select matricola from
                       Impiegati_attuali);
17
18                 IF FOUND THEN
19                         RAISE WARNING 'L IMPIEGATO CON MATRICOLA % E STATO
                               LICENZIATO, AGGIORNA IL REFERENTE SCIENTIFICO
                               NEL LABORATORIO %', NEW.matricola, impiegato.
                               id_lab;
20                 END IF;
21
22                 SELECT pa.referente, pa.cup INTO impiegato
23                 FROM PROGETTI_ATTUALI as pa
24                 WHERE NEW.matricola = pa.referente
25                 AND pa.referente NOT IN (select matricola from
                       Impiegati_attuali);
26
27                 IF FOUND THEN
28                         RAISE WARNING 'L IMPIEGATO CON MATRICOLA % E STATO
                               LICENZIATO, AGGIORNA IL REFERENTE NEL PROGETTO
                               %', NEW.matricola, impiegato.cup;
29                 END IF;
30
31                 SELECT pa.responsabile, pa.cup INTO impiegato
32                 FROM PROGETTI_ATTUALI as pa
33                 WHERE NEW.matricola = pa.responsabile
34                 AND pa.responsabile NOT IN (select matricola from
                       Impiegati_attuali);
35
36                 IF FOUND THEN
37                         RAISE WARNING 'L IMPIEGATO CON MATRICOLA % E STATO
                               LICENZIATO, AGGIORNA IL RESPONSABILE NEL
                               PROGETTO %', NEW.matricola, impiegato.cup;
38                 END IF;
39
40         END IF;
41
42         RETURN NEW;
43 END; $$ LANGUAGE plpgsql;
44
45 CREATE OR REPLACE TRIGGER trigger_avviso_su_impiegati_licenziati
46 AFTER UPDATE OF data_licenziamento ON impiegato
47 FOR EACH ROW
48 EXECUTE FUNCTION f_avviso_su_impiegati_licenziati();
```

6.2 Triggers su Laboratorio

- Il trigger **check_responsabile_scientifico** verifica in caso di inserimento o modifica del responsabile di un laboratorio che sia un senior come specificato da traccia, altrimenti ne impedisce l'inserimento o la modifica.

```
1  CREATE OR REPLACE TRIGGER check_responsabile_scientifico
2  BEFORE INSERT OR UPDATE OF r_scientifico ON laboratorio
3  FOR EACH ROW
4  EXECUTE FUNCTION check_responsabile_scientifico();
5
6  CREATE OR REPLACE FUNCTION f_check_responsabile_scientifico() RETURNS
       TRIGGER AS
7  $$
8  BEGIN
9          IF NEW.r_scientifico NOT IN (select matricola from Impiegati_attuali
                 where tipo_impiegato = 'senior') THEN
10                   RAISE EXCEPTION 'IL REFERENTE SCIENTIFICO DEVE ESSERE UN
                          SENIOR ATTUALE DELL AZIENDA!';
11         END IF;
12
13         RETURN NEW;
14 END;
15 $$ LANGUAGE plpgsql;
```

6.3 Triggers su Afferenza

- Il trigger **check_afferenza** verifica nel caso in cui voglio far afferire un impiegato ad un dato laboratorio inanzitutto che non sia licenziato, dopodichè controlla che l'ammontare delle ore lavorative al giorno siano minori di 8 ore.

```
1  CREATE OR REPLACE TRIGGER check_afferenza
2  AFTER INSERT OR UPDATE OF ore_giornaliere ON AFFERENZA
3  FOR EACH ROW
4  EXECUTE FUNCTION f_check_afferenza();
5
6  CREATE OR REPLACE FUNCTION f_check_afferenza() RETURNS TRIGGER AS
7  $$
8  BEGIN
9          IF(new.matricola not in(select matricola from Impiegati_attuali))
               THEN
10             RAISE EXCEPTION 'NON PUOI FAR AFFERIRE AD UN LABORATORIO UN
                    IMPIEGATO LICENZIATO';
11         END IF;
12
13         IF (new.ore_giornaliere > 8) THEN
14                 RAISE EXCEPTION 'UN IMPIEGATO NON PUO LAVORARE PER PIU DI
                        OTTO ORE AL GIORNO!';
15         END IF;
16         RETURN NEW;
17 END;
18 $$ LANGUAGE plpgsql;
```

- I seguenti trigger **inserisci_afferenti, cancella_afferenti** scattano all'inserimento o cancellazione di una o più tuple in afferenza, si occupano di aggiornare l'attributo numero_afferenti nella tabella laboratorio.

```
1  CREATE OR REPLACE TRIGGER inserisci_afferenti
2  AFTER INSERT ON AFFERENZA
3  FOR EACH ROW
4  EXECUTE FUNCTION f_inserisci_afferenti();
5
6  CREATE OR REPLACE TRIGGER cancella_afferenti
7  AFTER DELETE ON AFFERENZA
8  FOR EACH ROW
9  EXECUTE FUNCTION f_cancella_afferenti();
10
11 CREATE OR REPLACE FUNCTION f_inserisci_afferenti() RETURNS trigger AS
12 $$
13 BEGIN
14         UPDATE laboratorio
15         SET numero_afferenti = numero_afferenti + 1
16         WHERE id_lab = new.id_lab;
17 RETURN NEW;
18 END;
19 $$ Language plpgsql;
20
21 CREATE OR REPLACE FUNCTION f_cancella_afferenti() RETURNS trigger AS
22 $$
23 BEGIN
24         UPDATE laboratorio
25         SET numero_afferenti = numero_afferenti - 1
26         WHERE id_lab = old.id_lab;
27 RETURN NEW;
28 END; $$ LANGUAGE plpgsql;
```

6.4 Triggers su Progetto

- Questo trigger **check_update_dirigente_or_responsabile** scatta all'inserimento o modifica di una o più tuple nella tabella progetto, si occupa di verificare che il referente sia un dirigente e che il responsabile sia un senior, altrimenti blocca l'inserimento e manda un messaggio di errore.

```
1  CREATE OR REPLACE TRIGGER check_update_dirigente_or_responsabile
2  AFTER INSERT OR UPDATE OF referente, responsabile ON PROGETTO
3  FOR EACH ROW
4  EXECUTE FUNCTION f_check_referente_or_dirigente();
5
6  CREATE OR REPLACE FUNCTION f_check_referente_or_dirigente() RETURNS TRIGGER
       AS
7  $$
8  BEGIN
9          IF(new.responsabile not in (select matricola from Dirigenti_Attuali)
               ) THEN
10                 RAISE EXCEPTION 'IL RESPONSABILE DEVE ESSERE UN DIRIGENTE
                       NON LICENZIATO!';
11         END IF;
12         IF(new.referente not in (select matricola from Impiegati_attuali
               where tipo_impiegato = 'senior'))THEN
13                 RAISE EXCEPTION 'IL REFERENTE DEVE ESSERE UN IMPIEGATO
                       SENIOR NON LICENZIATO!';
14         END IF;
15 RETURN NEW;
16 END;
17 $$ language plpgsql;
```

6.5 Triggers su Gestione

- Il trigger **max_labs_per_cup** scatta all'inserimento di una o più tuple in gestione, si occupa di rispettare il vincolo imposto dalla traccia dove viene specificato che un progetto attivo ha al più tre laboratori associati.

```
1  CREATE OR REPLACE TRIGGER max_labs_per_cup
2  BEFORE INSERT ON GESTIONE
3  FOR EACH ROW
4  EXECUTE FUNCTION f_max_labs_per_cup();
5
6  CREATE OR REPLACE FUNCTION f_max_labs_per_cup() RETURNS TRIGGER AS
7  $$
8  DECLARE
9      lab_count INTEGER;
10 BEGIN
11         IF(new.cup in (select cup from PROGETTI_TERMINATI)) then
12                 RAISE EXCEPTION 'NON PUOI ASSOCIARE UN PROGETTO TERMINATO AD
                         UN LABORATORIO';
13         END IF;
14
15     SELECT COUNT(*) INTO lab_count FROM Gestione_Attuale WHERE cup = NEW.cup
           ;
16     IF lab_count > 3 THEN
17         RAISE EXCEPTION 'NON E POSSIBILE ASSOCIARE PIU DI TRE ID_LAB AD UN
               CUP (CODICE PROGETTO)';
18     END IF;
19     RETURN NEW;
20 END;
21 $$ LANGUAGE plpgsql;
```

7 Procedure

7.1 avviso_su_impiegati_licenziati

- La procedura **avviso_su_impiegati_licenziati** ha lo scopo di verificare quali impiegati sono arrivati al termine del contratto (sono giunti alla data_licenziamento) e , nel caso in cui vi sono degli impiegati con ruoli speciali, avvisa l'utente di aggiornare tali ruoli.

```
1  CREATE OR REPLACE PROCEDURE avviso_su_impiegati_licenziati() AS
2  $$ DECLARE
3  cursore_rscientifico CURSOR IS (select i.matricola, l.id_lab from impiegato
       AS i JOIN laboratorio AS l ON i.matricola = l.r_scientifico where i.
       matricola NOT IN (select matricola from Impiegati_attuali));
4  cursore_referenti CURSOR IS (select i.matricola, pa.cup from impiegato AS i
       JOIN PROGETTI_ATTUALI AS pa ON (i.matricola = pa.referente) where i.
       matricola NOT IN (select matricola from Impiegati_attuali));
5  cursore_responsabili CURSOR IS (select i.matricola, pa.cup from impiegato AS
        i JOIN PROGETTI_ATTUALI AS pa ON (i.matricola = pa.responsabile) where
        i.matricola NOT IN (select matricola from Impiegati_attuali));
6  rigacorrente RECORD;
7  BEGIN
8          OPEN cursore_rscientifico;
9          LOOP
10                 FETCH cursore_rscientifico INTO rigacorrente;
11                 IF NOT FOUND THEN
12                         EXIT;
13                 END IF;
14                 RAISE WARNING 'L IMPIEGATO CON MATRICOLA % E STATO
                       LICENZIATO, AGGIORNA IL REFERENTE SCIENTIFICO NEL
                       LABORATORIO %',
15                                                 rigacorrente.matricola,
                                                       rigacorrente.id_lab;
16         END LOOP;
17         CLOSE cursore_rscientifico;
18
19         OPEN cursore_referenti;
20         LOOP
21                 FETCH cursore_referenti INTO rigacorrente;
22                 IF NOT FOUND THEN
23                         EXIT;
24                 END IF;
25                 RAISE WARNING 'L IMPIEGATO CON MATRICOLA % E STATO
                       LICENZIATO, AGGIORNA IL REFERENTE NEL PROGETTO %',
26                                                 rigacorrente.matricola,
                                                       rigacorrente.cup;
27         END LOOP;
28         CLOSE cursore_referenti;
29
30         OPEN cursore_responsabili;
31         LOOP
32                 FETCH cursore_responsabili INTO rigacorrente;
33                 IF NOT FOUND THEN
34                         EXIT;
35                 END IF;
36                 RAISE WARNING 'L IMPIEGATO CON MATRICOLA % E STATO
                       LICENZIATO, AGGIORNA IL RESPONSABILE NEL PROGETTO %',
37                                                 rigacorrente.matricola,
                                                       rigacorrente.cup;
38         END LOOP;
39         CLOSE cursore_responsabili;
40 END; $$ LANGUAGE plpgsql;
```

7.2 update_database

- La procedura **update_database** è la routine di base per aggiornare automaticamente gli scatti di carriera di tutti gli impiegati, aggiornando anche l'attributo tipo_impiegato se necessario.

```
1  CREATE OR REPLACE PROCEDURE update_database() AS
2  $$ DECLARE
3          cursore_impiegati cursor is (select* from Impiegati_attuali as i
               natural join storico as s where s.nuovo_ruolo = i.
               tipo_impiegato);
4          imp_corrente record;
5  BEGIN
6          open cursore_impiegati;
7          LOOP
8          FETCH cursore_impiegati INTO imp_corrente;
9          IF NOT FOUND THEN
10                 EXIT;
11         end if;
12                 IF(imp_corrente.tipo_impiegato = 'junior') THEN
13                         IF(imp_corrente.data_assunzione + INTERVAL '3 years'
                                 <= CURRENT_DATE) THEN
14                           INSERT INTO STORICO VALUES('junior','middle'
                                 ,imp_corrente.data_assunzione +
                                 INTERVAL '3 YEARS', imp_corrente.
                                 matricola);
15                           IF(imp_corrente.data_assunzione + INTERVAL '
                                 7 years' >= CURRENT_DATE) THEN
16                              UPDATE IMPIEGATO
17                              SET tipo_impiegato = 'middle'
18                              WHERE impiegato.matricola =
                                     imp_corrente.matricola;
19                           ELSE
20                              INSERT INTO STORICO VALUES('middle',
                                     'senior',imp_corrente.
                                     data_assunzione + INTERVAL '7
                                     YEARS', imp_corrente.matricola)
                                     ;
21                              UPDATE IMPIEGATO
22                              SET tipo_impiegato = 'senior'
23                              WHERE impiegato.matricola =
                                     imp_corrente.matricola;
24                           END IF;
25                 END IF;
26                 END IF;
27                 IF(imp_corrente.tipo_impiegato = 'middle') THEN
28                         IF(imp_corrente.data_assunzione + INTERVAL '7 years'
                                 <= CURRENT_DATE) THEN
29                           INSERT INTO STORICO VALUES('middle','senior'
                                 ,imp_corrente.data_assunzione +
                                 INTERVAL '7 YEARS', imp_corrente.
                                 matricola);
30                           UPDATE IMPIEGATO
31                           SET tipo_impiegato = 'senior'
32                           WHERE impiegato.matricola = imp_corrente.
                                 matricola;
33                 END IF;
34                 END IF;
35         END LOOP;
36         close cursore_impiegati;
37 END; $$ LANGUAGE plpgsql;
```