SISTEMA DI GESTIONE DI PERSONALE E PROGETTI ALL'INTERNO DI UN'AZIENDA

Documentazione a cura di...

Valentina Piscopo
N86004086

Simone Scisciola
N86004025

Luca Piccolo
N86004407

Università degli Studi di Napoli Federico II

12 aprile 2023

Indice

Capitolo 0

Traccia

Sistema di gestione di personale e progetti di un'azienda

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:

1. Dipendente junior: colui che lavora da meno di 3 anni all'interno dell'azienda;
2. Dipendente middle: colui che lavora da meno di 7 ma da più di tre anni per l'azienda;
3. Dipendente senior: colui che lavora da almeno 7 anni per l'azienda.
4. 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. È necessario tracciare tutti gli scatti di carriera per ogni dipendente.
Nell'azienda vengono gestiti laboratori e progetti. Un laboratorio ha un particolare topic di cui si occupa, un certo numero di afferenti ed un responsabile scientifico che è un dipendente senior. 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. Al massimo 3 laboratori possono lavorare ad un progetto.

Per il gruppo da 3: Un laboratorio ha diverse attrezzature (computer, robot, dispositivi mobili, sensori, ...) acquistati con i fondi di un determinato progetto. Tracciare quindi gli acquisti fatti sui fondi di un progetto, con l'ovvio vincolo che il costo totale delle attrezzature non può superare il 50% del budget di un progetto. Con il restante 50% è possibile assumere dipendenti per l'azienda con un "contratto a progetto". Questi contratti hanno una scadenza, a differenza degli altri che sono a tempo indeterminato.

Capitolo 1

Requisiti identificati

1.1 Analisi dei requisiti

In questa sezione verranno illustrate le dinamiche chiave per la creazione di una base di dati atta alla gestione del personale e delle attività all'interno di un'azienda. L'azienda avrà due tipologie di dipendenti:

- Dipendenti con "contratto a progetto". Quest'ultimo possiede una data di scadenza, quindi il contratto sarà a tempo determinato;
- Dipendenti assunti stabilmente dall'azienda, con contratto a tempo indeterminato.
  Ciascun dipendente apparterrà, in base all'anzianità di servizio, ad una delle seguenti categorie:
  - dipendente "junior" se lavora da meno di 3 anni
  - dipendente "middle" se lavora da più di 3 anni ma meno di 7
  - dipendente "senior" se lavora da almeno 7 anni

Un dipendente con contratto a tempo indeterminato, a prescindere da quanto tempo lavori nell'azienda, può essere promosso a dirigente. In ogni momento, può anche essere rimosso da dirigente. Si vuole sviluppare uno storico degli scatti di carriera di ogni dipendente con contratto a tempo indeterminato.

Si vuole tenere traccia di laboratori e progetti gestiti nell'azienda.

Un laboratorio presenta:

- Un particolare topic di cui si occupa;
- Un responsabile scientifico, il quale deve essere un dipendente senior;
- Delle attrezzature.

Per ogni laboratorio, devono esistere un solo topic e un solo responsabile scientifico che lo coordina. Tuttavia, più laboratori potrebbero condividere topic o responsabile scientifico. Potrebbero esserci dipendenti che non sono responsabili scientifici.

Inoltre, si vuole tenere traccia del numero di afferenti ad un laboratorio. Un dipendente a tempo indeterminato potrebbe afferire a più laboratori così come potrebbe non afferire ad alcuno. Ad ogni modo, il laboratorio avrà sicuramente almeno un afferente, cioè il responsabile scientifico per quel laboratorio.

Un laboratorio potrebbe non aver lavorato ad alcun progetto, così come potrebbe aver lavorato su più progetti. Similmente, potrebbe avere delle attrezzature così come potrebbe non averne alcuna.

Un progetto ha diverse caratteristiche:

- Un CUP (codice unico progetto);
- Un nome, il quale è unico nel sistema;

- Un referente scientifico, il quale deve essere un dipendente senior;
- Un responsabile, il quale è un dirigente;
- I fondi, che finanziano il progetto, vincolati solo e unicamente a quel progetto.

Per ogni progetto devono esistere e sono unici un referente scientifico ed un responsabile. Ciò nonostante, più progetti potrebbero condividere referente scientifico o responsabile. Un dipendente può anche non ricoprire questi ruoli.

Un progetto potrebbe non essere preso in carico da nessun laboratorio, così come potrebbe essere assegnato a uno o più laboratori, fino ad un massimo di tre.

Tramite i fondi di un progetto possono essere acquistate delle attrezzature di cui il progetto sarà proprietario (ad esempio, computer, robot, dispositivi mobili, sensori ...), le quali possono essere assegnate ad un laboratorio che ha lavorato al progetto. Si intende tenere traccia di tali acquisti. Le attrezzature acquistate potrebbero anche non essere utilizzate da alcun laboratorio. Nel caso, però, venissero assegnate ad un laboratorio in particolare, questo deve essere uno dei laboratori che hanno lavorato al progetto tramite cui è stata acquistata l'attrezzatura.

I fondi di un progetto possono essere utilizzati unicamente per quel progetto e, se un progetto esiste, significa che è stato ammesso a finanziamento. Dunque, non possono esserci progetti senza fondi. Inoltre:

- Non oltre il 50% dei fondi può essere destinato all'acquisto delle attrezzature. Malgrado ciò, è anche possibile non acquistare alcuna attrezzatura tramite i fondi di un progetto.
- Non oltre il restante 50% dei fondi è da destinare ai dipendenti assunti con un "contratto a progetto" che lavoreranno su questo progetto. L'esistenza di un dipendente con "contratto a progetto" implica che sia stato assunto con i fondi di quel progetto. Comunque, è anche possibile non assumere alcun dipendete con "contratto a progetto" con i fondi di un progetto.

Sia l'acquisto di una particolare attrezzatura che l'assunzione di un particolare dipendente con "contratto a progetto" può essere fatto solo con i fondi di un singolo progetto. Ovvero, non è possibile acquistare la stessa attrezzatura, o ingaggiare lo stesso dipendente, con fondi di più progetti diversi. Un dipendente con "contratto a progetto" non può lavorare su più progetti (o non lavorare ad alcuno) poiché è stato ingaggiato per lavorare a quel progetto specifico.

1.2 Scelte progettuali

Vengono introdotti di seguito alcuni attributi che solitamente si intende tracciare in una base dati, come le generalità di un dipendente, il nome di un laboratorio o di un'attrezzatura, una data di inizio e fine progetto.

In particolare, per quanto riguarda i dipendenti, introduciamo un attributo "matricola" che rappresenta, mediante l'apposito contratto, l'impegno lavorativo del dipendente presso l'azienda. Se, ad esempio, un dipendente venisse licenziato e poi riassunto, gli verrebbe assegnata una nuova matricola data dalla stipulazione di un nuovo contratto. Nel caso in cui un dipendente con contratto a tempo indeterminato venisse licenziato, e successivamente riassunto, partirà dalla classificazione "junior", dal momento che, con la stipulazione di un nuovo contratto, si otterrà un nuovo periodo di lavoro. Nonostante ciò, verrà comunque tenuta traccia nel database dei dati riguardanti la carriera del dipendente prima di essere licenziato.

Anche i dipendenti con "contratto a progetto" saranno identificati da una matricola che verrà riassegnata ad ogni nuovo contratto, tenendo comunque traccia delle attività precedenti nel caso di vecchi contratti a progetto. Introduciamo una data di assunzione che definirà l'apertura del contratto, che non dovrà necessariamente corrispondere alla data in cui il progetto ha inizio. Tuttavia, la data di scadenza del contratto non potrà superare la data in cui il progetto ha effettivamente fine.

Capitolo 2

Progettazione concettuale

2.1 Diagramma ER

Dall'analisi dei requisiti costruiamo il diagramma ER.
In particolare, il modello concettuale basato sull'analisi dei requisiti permetterà di creare una base di dati efficace per la gestione del personale e delle attività svolte nell'azienda.
Il modello dovrà tenere in considerazione le due tipologie di dipendenti e le relative informazioni tracciate nella base di dati, i laboratori, i progetti, i fondi e gli acquisti effettuati tramite i fondi.

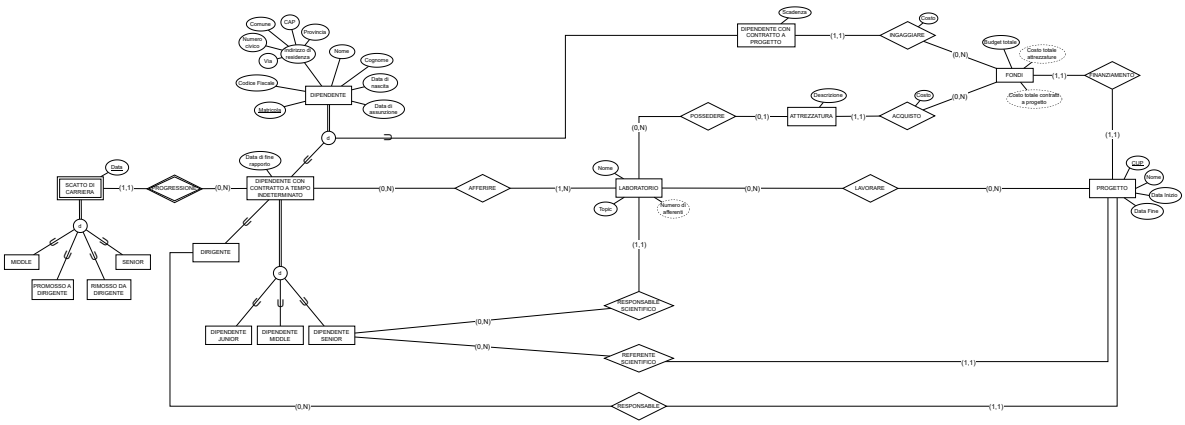

Figura 2.1: Diagramma ER non ristrutturato

Capitolo 3

Ristrutturazione della progettazione concettuale

3.1 Analisi della ristrutturazione del Diagramma ER

3.1.1 Analisi delle ridondanze

Nel modello concettuale sono presenti attributi ridondanti. Tali attributi sono "Numero di afferenti" dell'entità "LABORATORIO", "Costo totale attrezzature" e "Costo totale contratti a progetto" dell'entità "PROGETTO", poiché attributi derivabili. Per risolvere le ridondanze, analizziamo separatamente il caso in cui scegliamo di mantenere la ridondanza e il caso in cui scegliamo di rimuoverla. Supponiamo di eseguire le seguenti operazioni:

- Registrazione delle afferenze di un dipendete indeterminato a uno o più laboratori. Visualizzazione di tutti i dati relativi a un laboratorio, circa 5 volte al mese.
- Acquisto di una nuova attrezzatura, indicandone il costo e il progetto tramite cui effettuiamo l'acquisto. Visualizzazione di tutti i dati relativi ad un progetto, circa 25 volte al mese.
- Assunzione di un nuovo dipendente con contratto a progetto, indicandone il costo del contratto e il progetto per cui lavorerà. Visualizzazione di tutti i dati relativi ad un progetto, circa 25 volte al mese, come nel caso delle attrezzature.

Tenendo conto del costo doppio degli accessi in scrittura rispetto agli accessi in lettura ed un uso di memoria molto ridotto a supporto degli attributi ridondanti, si può concludere che:

- Nel tipo di entità "LABORATORIO", con l'inserimento dell'attributo ridondante "Numero di afferenti", si ottiene un risparmio di circa la metà degli accessi mensili al database, considerando le operazioni precedentemente riportate.
- Nel tipo di entità "PROGETTO", con l'inserimento dell'attributo ridondante "Costo totale attrezzature", si ottiene un risparmio di circa poco meno della metà degli accessi mensili al database per le operazioni precedentemente riportate.
- Nel tipo di entità "PROGETTO", con l'inserimento dell'attributo ridondante "Costo totale contratti a progetto", si ottiene un risparmio di più della metà degli accessi mensili al database per le operazioni precedentemente riportate.

E' presente un'ulteriore ridondanza tra la sottoclasse "DIRIGENTE" della superclasse "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" e la sottoclasse "PROMOSSO A DIRIGENTE" dalla specializzazione dell'entità debole "SCATTO DI CARRIERA". Tale ridondanza viene mantenuta, facendo riferimento ai criteri già illustrati, per garantire una facilità di accesso ai dati ed un miglioramento delle prestazioni, anche se minimo.

Inoltre, la relazione "LAVORARE" tra un laboratorio ed un progetto potrebbe sembrare una ridondanza, dato che se un laboratorio ha un'attrezzatura acquistata tramite i fondi di un progetto, di certo avrà lavorato a quel progetto. Tuttavia, senza la relazione "LAVORARE" non sarebbe possibile tenere traccia di eventuali laboratori che hanno lavorato a un progetto ma a cui non è stato assegnato alcuna attrezzatura acquistata tramite i fondi del progetto. In effetti, si potrebbe anche non acquistare

alcuna attrezzatura tramite tali fondi.

3.1.2 Analisi degli identificativi

Per ognuno dei seguenti tipi di entità scegliamo gli identificativi:

- DIPENDENTE
  Scegliamo come identificativo l'attributo "Matricola", il quale assumiamo essere un codice alfanumerico di otto caratteri. Ad ogni dipendente sarà assegnata univocamente una matricola al momento dell'assunzione. Ogni matricola sarà diversa dalle altre ed identificherà un dipendente che lavora all'interno dell'azienda grazie al contratto di assunzione. Se il dipendente venisse licenziato e in seguito riassunto, riceverebbe una nuova matricola dovuta alla nuova assunzione con un contratto diverso.

- PROGETTO
  Scegliamo come identificativo l'attributo "CUP". Il CUP è un codice composto da quindici caratteri alfanumerici atto ad identificare univocamente un progetto.

- LABORATORIO
  Scegliamo come identificativo l'attributo "Nome", dato che all'interno di una singola azienda assumiamo che non venga assegnato ad un laboratorio un nome precedentemente assegnato ad altri laboratori.

- ATTREZZATURA
  Introduciamo come identificativo un attributo "idAttrezzatura", ovvero un codice numerico che identifica ogni attrezzatura acquistata.

3.1.3 Rimozione degli attributi multivalore

Non abbiamo modellato alcuna situazione presentante attributi multivalore.

3.1.4 Rimozione degli attributi composti

L'unico attributo composto presente nel diagramma concettuale è l'attributo di "DIPENDENTE" chiamato "Indirizzo". Scegliamo di trattare tale attributo come attributo atomico, trascurando i vari attributi componenti semplici. Infatti supponiamo che l'azienda in questione avrà di rado bisogno di fare ricerche basate sugli indirizzi (oppure su informazioni ricavabili da essi), dato che la base di dati è modellata principalmente per la gestione dei laboratori, progetti e dipendenti di un'azienda.

3.1.5 Partizione/Accorpamento delle associazioni

Dal momento i tipi di entità "FONDI" e "PROGETTO" sono coinvolti nell'associazione "FINANZIAMENTO" con cardinalità 1:1 e entrambi presentano partecipazione totale, accorpiamo i due tipi di entità in "PROGETTO" in modo da accedere ai fondi che finanziano un progetto (o al progetto che viene finanziato da dei fondi) senza attraversare un'ulteriore relazione "FINANZIAMENTO".
Non sono presenti entità da partizionare.

3.1.6 Rimozione delle gerarchie

Nel modello concettuale troviamo diverse gerarchie da rimuovere. In particolare:

- Viene rimossa la specializzazione dell'entità debole "SCATTI DI CARRIERA" - "MIDDLE" / "SENIOR" / "PROMOSSO A DIRIGENTE" / "RIMOSSO DA DIRIGENTE". Si ricorre ad una relazione singola con attributo di tipo o discriminante, dal momento che le sottoclassi non presentano alcuna distinzione in quanto ad attributi. Dunque, si introduce il nuovo attributo "Tipo", che rispecchia il tipo di scatto che un dipendente effettua in una certa data.

- Viene rimossa la specializzazione "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" - "JUNIOR" / "MIDDLE" / "SENIOR", utilizzando una relazione singola con attributo discriminante. Dunque, si inserisce il nuovo attributo "Tipo" che specifica l'anzianità del dipendente.
  Le sottoclassi "DIPENDENTE JUNIOR", "DIPENDENTE MIDDLE" e "DIPENDENTE SENIOR" non differiscono per alcun attributo dato che la distinzione è puramente semantica, basata sul tempo di attività in azienda.
  In questo modo, viene limitato alla sola superclasse il numero di associazioni da creare che, altrimenti, sarebbero ripetute per ogni sottoclasse. Inoltre, siccome non vi sono attributi specifici delle singole sottoclassi, eviteremo anche di avere interi attributi nulli per determinati tipi di dipendente.
  Però, a causa del nuovo attributo, sarà presente una ridondanza riguardante gli scatti di carriera. Infatti, l'informazione circa l'anzianità del dipendente è reperibile sia nel tipo di entità "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO", sia nel tipo di entità debole "SCATTI DI CARRIERA". Si decide di mantenere tale ridondanza per favorire una facilità di accesso ai dati ed un miglioramento delle prestazioni, anche se minimo.

- Riguardo la sottoclasse "DIRIGENTE", si introduce nella sua superclasse "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" un nuovo attributo "DIRIGENTE", per segnalare un eventuale ruolo dirigenziale del dipendente, conseguibile a prescindere dall'anzianità di servizio.

- Viene rimossa la specializzazione "DIPENDENTE" - "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" / "DIPENDENTE CON CONTRATTO A PROGETTO", utilizzando due relazioni di sottoclasse, dal momento che la specializzazione è totale e disgiunta.

3.2 Diagramma ER ristrutturato

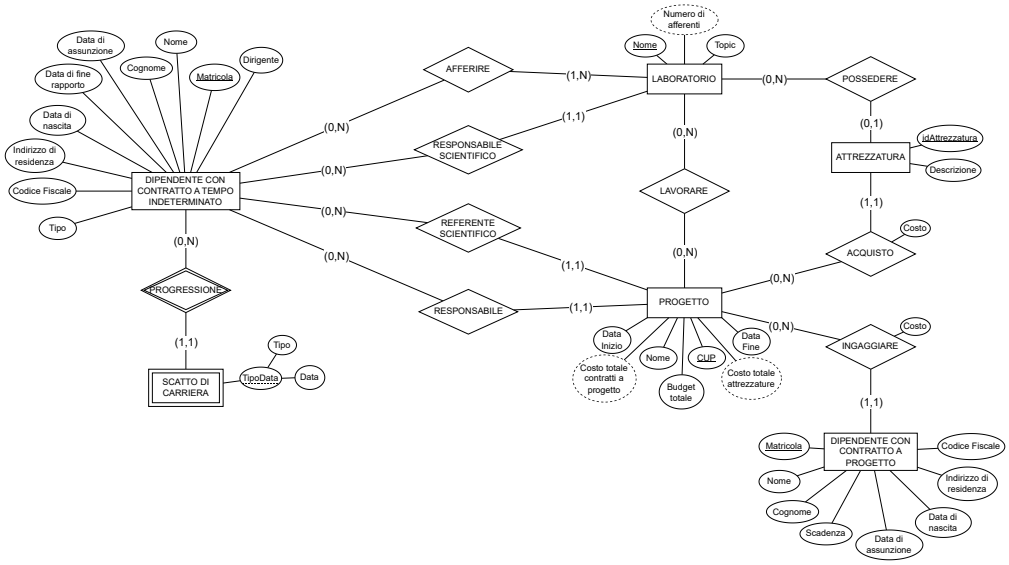

Figura 3.1: Diagramma ER ristrutturato

3.3 Class Diagram ristrutturato

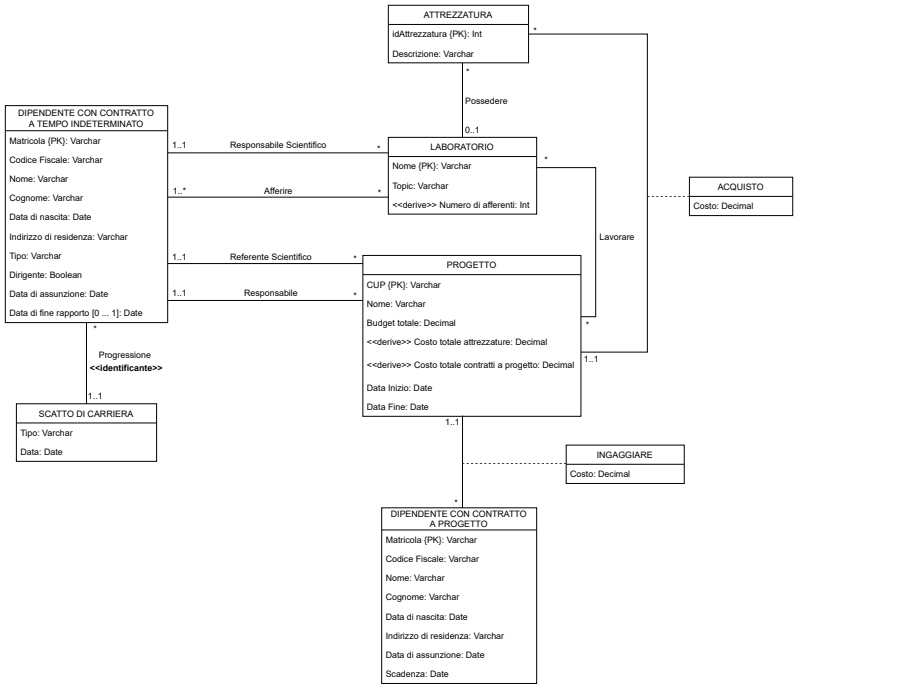

Figura 3.2: Class Diagram ristrutturato

3.4 Dizionario delle entità

Entità	Descrizione
DIPENDENTE CON CONTRATTO A TEMPO INDE-TERMINATO	Il tipo di entità "Dipendente con contratto a tempo indeterminato" rappresenta i dipendenti assunti con contratto a tempo indeterminato dall'azienda. Un dipendente con contratto a tempo indeterminato avrà caratteristiche quali: la matricola (unica per ogni dipendente), il nome, il cognome, il codice fiscale, l'indirizzo di residenza, la data di nascita, la data di fine rapporto (ovvero la data in cui eventualmente smette di prestare servizio nell'azienda), la data di assunzione, un attributo tipo che specifica l'anzianità del dipendente (ovvero se è un dipendente junior, middle o senior) e un attributo che indica se il dipendente è attualmente un dirigente o meno.
SCATTO DI CARRIERA	Il tipo di entità debole "Scatto di carriera" rappresenta gli scatti di carriera di un dipendente. In particolare lo scatto può essere di quattro tipi: 1. Scatto da junior a middle, indicato con "Middle" 2. Scatto da middle a senior, indicato con "Senior" 3. Scatto da non dirigente a dirigente, che indichiamo con la denominazione "Promosso a dirigente", non vincolato all'anzianità 4. Scatto da dirigente a non dirigente, che indichiamo con la denominazione "Rimosso da dirigente", non vincolato all'anzianità. Oltre al tipo di scatto avremo anche la data in cui è avvenuto lo scatto per una determinata matricola.
LABORATORIO	Il tipo di entità "Laboratorio" rappresenta i laboratori che si trovano attualmente all'interno dell'azienda. In particolare un laboratorio avrà un nome unico nell'azienda, un topic ed un valore numerico che indica il numero di dipendenti che ad esso afferiscono.
ATTREZZATURA	Il tipo di entità "Attrezzatura" rappresenta le attrezzature acquistate tramite i fondi di progetti, che possono o meno trovarsi all'interno di un laboratorio. In particolare, un'attrezzatura è un oggetto che avrà un identificativo ed una descrizione (che appunto descrive l'oggetto in questione).
PROGETTO	Il tipo di entità "Progetto" rappresenta i progetti presi in carico dall'azienda. In particolare il progetto possiederà un CUP (codice unico progetto), un nome, una data di inizio e di fine esecuzione del progetto, il budget totale (che rappresenta i fondi istanziati per quel progetto), il costo totale delle attrezzature acquistate tramite i fondi di quel progetto ed il costo totale dei contratti a progetto dei dipendenti assunti tramite i fondi di quel progetto
DIPENDENTE CON "CONTRATTO A PROGETTO"	Il tipo di entità "Dipendente con contratto a progetto" rappresenta i dipendenti assunti per lavorare su un progetto tramite un contratto a tempo determinato. Un dipendente con "contratto a progetto" presenterà: una matricola (unica per ogni dipendente), un nome, un cognome, un codice fiscale, un indirizzo di residenza, una data di nascita, una data di assunzione, una data di scadenza (ovvero la data in cui scade il contratto a tempo determinato).

3.5 Dizionario delle associazioni

Associazione	Descrizione
PROGRESSIONE	L'associazione "PROGRESSIONE" tra "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" e "SCATTO DI CARRIERA" è un'associazione identificante il tipo di entità debole "SCATTO DI CARRIERA", utile per risalire agli scatti di carriera di ogni dipendente. In particolare, un dipendente può avere diversi scatti di carriera, o nessuno. Viceversa, uno scatto è relativo ad un unico dipendente.
RESPONSABILE SCIENTIFICO	L'associazione "RESPONSABILE SCIENTIFICO" tra "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" e "LABORATORIO" specifica quali dipendenti sono responsabili scientifici di un laboratorio. In particolare, un dipendete potrebbe essere, o meno, Responsabile scientifico di uno o più laboratori. Viceversa, un laboratorio avrà uno ed un solo Responsabile scientifico.
AFFERIRE	"AFFERIRE" tra "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" e "LABORATORIO" indica quali sono i dipendenti che afferiscono attualmente ad un particolare laboratorio. In particolar modo, un dipendente può afferire a più laboratori, ma potrebbe anche non afferirne a nessuno. Viceversa, un laboratorio può avere più afferenti, oltre al responsabile scientifico.
REFERENTE SCIENTIFICO	L'associazione "REFERENTE SCIENTIFICO" tra "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" e "PROGETTO" denota quali dipendenti sono referenti scientifici di un progetto. In particolare, un dipendente potrebbe essere, o meno, referente scientifico di uno o più progetti. Al contrario, un progetto deve avere uno ed un solo referente scientifico.
RESPONSABILE	"RESPONSABILE" tra "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" e "PROGETTO" indica quali dipendenti sono responsabili di un progetto. In particolare, un dipendete potrebbe essere, o meno, responsabile di uno o più progetti, mentre un progetto avrà uno ed un solo responsabile.
LAVORARE	L'associazione "LAVORARE" tra "LABORATORIO" e "PROGETTO" fa corrispondere, ad ogni laboratorio, i progetti a cui ha lavorato. In particolare, un laboratorio può aver lavorato a più progetti, così come potrebbe non aver mai lavorato ad alcun progetto. Viceversa, un progetto presenta dei laboratori che lavorano ad esso, oppure nessuno.
POSSEDERE	"POSSEDERE" tra "LABORATORIO" e "ATTREZZATURA" indica le attrezzature appertenenti ad un laboratorio. Un laboratorio potrebbe avere più attrezzature, così come potrebbe non averne nessuna. All'opposto, un'attrezzatura appartiene, o meno, ad un laboratorio.
ACQUISTO	"ACQUISTO" tra "ATTREZZATURA" e "PROGETTO" fornisce informazioni circa le attrezzature acquistate tramite i fondi di un progetto, infatti ogni acquisto avrà un costo. Dunque, sarà possibile acquistare più attrezzature, o nessuna, per un progetto. Invece, tutte le attrezzature devono essere acquistate tramite i fondi di uno ed un solo progetto.
INGAGGIARE	L'associazione tra "PROGETTO" e "DIPENDENTE CON CONTRATTO A PROGETTO" denominata "INGAGGIARE" indica quali sono i dipendenti ingaggiati utilizzando il budget di un progetto, ed ogni ingaggio di un nuovo contratto avrà un costo. Infatti, sarà possibile ingaggiare più dipendenti con contratto a progetto, o nessuno. Viceversa, un dipendente con contratto a progetto, se presente, è stato assunto mediante i fondi di uno ed un solo progetto.

Capitolo 4

Schema logico

4.1 Correzione e abbreviazioni dei nomi

Costruendo lo schema logico, per facilitare la traduzione di tipi di entità e associazioni, abbiamo deciso di rinominare alcuni tipi di entità e associazioni, in modo da abbreviarne il nome e togliere eventuali spazi, in vista dello schema fisico. Di seguito la lista di tutte le modifiche apportate. Rinominiamo:

- "DIPENDENTE CON CONTRATTO A TEMPO INDETERMINATO" in "DIP_INDETERMINATO"
  - "Codice Fiscale" in "codFiscale"
  - "Indirizzo di residenza" in "Indirizzo"
  - "Data di nascita" in "dataNascita"
  - "Data di assunzione" in "dataAssunzione"
  - "Data di fine rapporto" in "dataFine"
- "DIPENDENTE CON CONTRATTO A PROGETTO" in "DIP_PROGETTO"
  - "Codice Fiscale" in "codFiscale"
  - "Indirizzo di residenza" in "Indirizzo"
  - "Data di nascita" in "dataNascita"
  - "Data di assunzione" in "dataAssunzione"
- "SCATTO DI CARRIERA" in "SCATTO_CARRIERA"
- "RESPONSABILE SCIENTIFICO" in "RESPONSABILE_SCIENTIFICO"
- "REFERENTE SCIENTIFICO" in "REFERENTE_SCIENTIFICO"
- "LABORATORIO"
  - "Numero di afferenti" in "nAfferenti"
- "PROGETTO"
  - "Data inizio" in "dataInizio"
  - "Data fine" in "dataFine"
  - "Budget totale" in "Budget"
  - "Costo totale contratti a progetto" in "costoContrattiProgetto"
  - "Costo totale attrezzature" in "costoAttrezzature"

4.2 Mappatura dei tipi di associazioni

Di seguito vengono discussi i criteri adottati per la mappatura delle varie tipologie di associazioni.

- Mappatura di tipi di associazioni identificanti:

  1. Associazione "PROGRESSIONE":
     Nel tipo di entità debole "SCATTO CARRIERA" viene inserito l'attributo di chiave esterna "Matricola", che fa riferimento all'attributo di chiave primaria "Matricola" del tipo di entità "DIP_INDETERMINATO". La chiave parziale del tipo di entità debole "SCATTO CARRIERA" sarà composta dal nuovo attributo di chiave esterna "Matricola" e dagli attributi "Tipo" e "Data", che formavano la precedente chiave parziale.

- Associazioni binarie 1:N:
  Per ogni associazione riportata, viene utilizzato l'approccio ***basato su chiavi esterne***.

  1. Associazione "RESPONSABILE SCIENTIFICO":
     Nel tipo di entità "LABORATORIO", avente cardinalità "N" rispetto l'associazione, viene inserito l'attributo di chiave esterna "Responsabile_Scientifico", che farà riferimento all'attributo di chiave primaria "Matricola" del tipo di entità "DIP_INDETERMINATO".
  2. Associazione "REFERENTE SCIENTIFICO":
     Nel tipo di entità "PROGETTO", avente cardinalità "N" rispetto l'associazione, viene inserito l'attributo di chiave esterna "Referente_Scientifico", che farà riferimento all'attributo di chiave primaria "Matricola" del tipo di entità "DIP_INDETERMINATO".
  3. Associazione "RESPONSABILE":
     Nel tipo di entità "PROGETTO", avente cardinalità "N" rispetto l'associazione, viene inserito l'attributo di chiave esterna "Responsabile", che farà riferimento all'attributo di chiave primaria "Matricola" del tipo di entità "DIP_INDETERMINATO".
  4. Associazione "POSSEDERE":
     Nel tipo di entità "ATTREZZATURA", avente cardinalità "N" rispetto l'associazione, viene inserito l'attributo di chiave esterna "nomeLab", che farà riferimento all'attributo di chiave primaria "Nome" del tipo di entità "LABORATORIO".
  5. Associazione "ACQUISTO":
     Nel tipo di entità "ATTREZZATURA", avente cardinalità "N" rispetto l'associazione, viene inserito l'attributo di chiave esterna "CUP", che farà riferimento all'attributo di chiave primaria "CUP" del tipo di entità "PROGETTO". Inoltre, nel tipo di entità, verrà inserito l'attributo "Costo" dell'associazione in questione.
  6. Associazione "INGAGGIARE":
     Nel tipo di entità "DIP_PROGETTO", avente cardinalità "N" rispetto l'associazione, viene inserito l'attributo di chiave esterna "CUP", che farà riferimento all'attributo di chiave primaria "CUP" del tipo di entità "PROGETTO". Inoltre, nel tipo di entità, verrà inserito l'attributo "Costo" dell'associazione in questione.

- Associazioni binarie M:N:
  Per ogni associazione riportata, viene utilizzato l'approccio ***basato su relazione di relazioni***.

  1. Associazione "AFFERIRE":
     Viene creata una nuova relazione "AFFERIRE" composta dai seguenti attributi di chiave esterna: "Matricola", che fa riferimento all'attributo di chiave primaria "Matricola" del tipo di entità "DIP_INDETERMINATO", e "nomeLab", che fa riferimento all'attributo di chiave primaria "Nome" del tipo di entità "LABORATORIO".
     La chiave primaria della nuova relazione sarà formata da entrambi gli attributi di chiave esterna.
  2. Associazione "LAVORARE":
     Viene creata una nuova relazione "LAVORARE" composta dai seguenti attributi di chiave esterna: "CUP", che fa riferimento all'attributo di chiave primaria "CUP" del tipo di entità "PROGETTO", e "nomeLab", che fa riferimento all'attributo di chiave primaria "Nome" del tipo di entità "LABORATORIO".
     La chiave primaria della nuova relazione sarà composta da entrambi gli attributi di chiave esterna.

4.3 Schema logico

- DIP_INDETERMINATO(Matricola, Tipo, Nome, Cognome, codFiscale, Indirizzo, dataNascita, dataAssunzione, dataFine, Dirigente)
- SCATTO_CARRIERA(Matricola, Tipo, Data)

  SCATTO_CARRIERA.Matricola → DIP_INDETERMINATO.Matricola
- LABORATORIO(Nome, Topic, nAfferenti, Responsabile_Scientifico)

  LABORATORIO.Responsabile_Scientifico → DIP_INDETERMINATO.Matricola
- AFFERIRE(Matricola, nomeLab)

  AFFERIRE.Matricola → DIP_INDETERMINATO.Matricola

  AFFERIRE.nomeLab → LABORATORIO.Nome
- PROGETTO(CUP, Nome, dataInizio, dataFine, Budget, costoAttrezzature, costoContrattiProgetto, Referente_Scientifico, Responsabile)

  PROGETTO.Referente_Scientifico → DIP_INDETERMINATO.Matricola

  PROGETTO.Responsabile → DIP_INDETERMINATO.Matricola
- LAVORARE(CUP, nomeLab)

  LAVORARE.CUP → PROGETTO.CUP

  LAVORARE.nomeLab → LABORATORIO.Nome
- ATTREZZATURA(idAttrezzatura, Descrizione, nomeLab, CUP, Costo)

  ATTREZZATURA.nomeLab → LABORATORIO.Nome

  ATTREZZATURA.CUP → PROGETTO.CUP
- DIP_PROGETTO(Matricola, Nome, Cognome, codFiscale, Indirizzo, dataNascita, dataAssunzione, Scadenza, CUP, Costo)

  DIP_PROGETTO.CUP → PROGETTO.CUP

Capitolo 5

Schema fisico

5.1 Creazione schema

```
CREATE SCHEMA azienda AUTHORIZATION postgres;
```

5.2 Creazione domini

```
CREATE DOMAIN azienda.STRING AS VARCHAR(30);


CREATE DOMAIN azienda.EURO AS NUMERIC(20, 2)
    CONSTRAINT POSITIVE_EURO
    CHECK (VALUE >= 0);


CREATE DOMAIN azienda.MATRICOLA AS VARCHAR(8);
    CONSTRAINT DOM_MATRICOLA_CHECK_LENGTH
    CHECK (LENGTH(VALUE) = 8);


CREATE DOMAIN azienda.CUP AS VARCHAR(15);
    CONSTRAINT DOM_CUP_CHECK_LENGTH
    CHECK (LENGTH(VALUE) = 15);
    CONSTRAINT DOM_CUP_CHECK_ALPHANUMERIC
    CHECK (VALUE ~'[[:alnum:]]15');


CREATE DOMAIN azienda.CODFISCALE AS VARCHAR(16)
    CONSTRAINT DOM_CODFISCALE_CHECK_LENGTH
    CHECK (LENGTH(VALUE) = 16);
    CONSTRAINT DOM_CODFISCALE_ALPHANUMERIC
    CHECK (VALUE ~'[[:alnum:]]16');


CREATE DOMAIN azienda.TIPO_DIPENDENTE AS VARCHAR(6)
    CONSTRAINT DOM_TIPO_DIPENDENTE_CHECK_JMS
    CHECK (UPPER(VALUE) IN ('JUNIOR', 'MIDDLE', 'SENIOR'));


CREATE DOMAIN azienda.TIPO_SCATTO AS VARCHAR(20)
    CONSTRAINT DOM_TIPO_SCATTO_CHECK_MSD
    CHECK (UPPER(VALUE) IN ('MIDDLE', 'SENIOR', 'PROMOSSO A DIRIGENTE', 'RIMOSSO DA
    DIRIGENTE'));
```

5.3 Creazione tabelle

5.3.1 DIP_INDETERMINATO

```
CREATE TABLE azienda.DIP_INDETERMINATO(

     Matricola azienda.MATRICOLA NOT NULL,
     Tipo azienda.TIPO_DIPENDENTE NOT NULL DEFAULT 'Junior',
     Nome azienda.STRING NOT NULL,
     Cognome azienda.STRING NOT NULL,
     codFiscale azienda.CODFISCALE NOT NULL,
     Indirizzo VARCHAR(100),
     dataNascita DATE NOT NULL,
     dataAssunzione DATE NOT NULL DEFAULT FALSE,
     dataFine DATE,
     Dirigente BOOLEAN NOT NULL,

     CONSTRAINT pk_dip_indeterminato PRIMARY KEY (Matricola),
     CONSTRAINT check_ordine_date_di CHECK (dataNascita < dataAssunzione AND
        dataAssunzione <= dataFine)

    );
```

5.3.2 SCATTO_CARRIERA

```
CREATE TABLE azienda.SCATTO_CARRIERA(

     Matricola azienda.MATRICOLA NOT NULL,
     Tipo azienda.TIPO_SCATTO NOT NULL,
     Data DATE NOT NULL,

     CONSTRAINT pk_scatto_carriera PRIMARY KEY (Matricola, Tipo, Data),
     CONSTRAINT fk_matricola_scatto_carriera FOREIGN KEY (Matricola)
        REFERENCES azienda.dip_indeterminato(Matricola)
        ON DELETE CASCADE ON UPDATE CASCADE

    );
```

5.3.3 LABORATORIO

```
CREATE TABLE azienda.LABORATORIO(

    Nome azienda.STRING NOT NULL,
    Topic azienda.STRING NOT NULL,
    nAfferenti INTEGER NOT NULL DEFAULT 1,
    Responsabile_Scientifico azienda.MATRICOLA NOT NULL,

    CONSTRAINT pk_laboratorio PRIMARY KEY (Nome),
    CONSTRAINT fk_responsabile_scientifico_laboratorio FOREIGN KEY
       (Responsabile_Scientifico)
       REFERENCES azienda.dip_indeterminato(Matricola)
       ON DELETE NO ACTION ON UPDATE CASCADE,
    CONSTRAINT check_positive_nAfferenti CHECK (nAfferenti > 0)

   );
```

5.3.4 AFFERIRE

```
CREATE TABLE azienda.AFFERIRE(

    Matricola azienda.MATRICOLA NOT NULL,
    nomeLab azienda.STRING NOT NULL,

    CONSTRAINT pk_afferire PRIMARY KEY (Matricola, nomeLab),
    CONSTRAINT fk_matricola_afferire FOREIGN KEY (Matricola)
       REFERENCES azienda.dip_indeterminato(Matricola)
       ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_nome_afferire FOREIGN KEY (nomeLab)
       REFERENCES azienda.laboratorio(Nome)
       ON DELETE CASCADE ON UPDATE CASCADE

   );
```

5.3.5 PROGETTO

```
CREATE TABLE azienda.PROGETTO(

    CUP azienda.CUP NOT NULL,
    Nome azienda.STRING,
    dataInizio DATE NOT NULL,
    dataFine DATE,
    Budget azienda.EURO NOT NULL,
    costoAttrezzature azienda.EURO NOT NULL DEFAULT 0,
    costoContrattiProgetto azienda.EURO NOT NULL DEFAULT 0,
    Referente_Scientifico azienda.MATRICOLA NOT NULL,
    Responsabile azienda.MATRICOLA NOT NULL,

    CONSTRAINT pk_progetto PRIMARY KEY (CUP),
    CONSTRAINT fk_referente_scientifico_progetto FOREIGN KEY
       (Referente_Scientifico)
       REFERENCES azienda.dip_indeterminato(Matricola)
       ON DELETE NO ACTION ON UPDATE CASCADE,
    CONTRAINT fk_responsabile_scientifico_progetto FOREIGN KEY (Responsabile)
       REFERENCES azienda.dip_indeterminato(Matricola)
       ON DELETE NO ACTION ON UPDATE CASCADE,
    CONSTRAINT nome_progetto_unico UNIQUE(Nome),
    CONSTRAINT check_positive_budget CHECK (Budget > 0),
    CONSTRAINT check_date_fine_inizio CHECK (dataInizio <= dataFine)

   );
```

5.3.6 LAVORARE

```
CREATE TABLE azienda.LAVORARE(

    CUP azienda.CUP NOT NULL,
    nomeLab azienda.STRING NOT NULL,

    CONSTRAINT pk_lavorare PRIMARY KEY (CUP, nomeLab),
    CONSTRAINT fk_cup_lavorare FOREIGN KEY (CUP)
       REFERENCES azienda.progetto(CUP)
       ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_nome_lavorare FOREIGN KEY (nomeLab)
       REFERENCES azienda.laboratorio(Nome)
       ON DELETE CASCADE ON UPDATE CASCADE

   );
```

5.3.7 ATTREZZATURA

```
CREATE TABLE azienda.ATTREZZATURA(

     idAttrezzatura SERIAL NOT NULL,
     Descrizione VARCHAR(256) NOT NULL,
     Costo azienda.EURO NOT NULL,
     nomeLab azienda.STRING,
     CUP azienda.CUP NOT NULL,

     CONSTRAINT pk_attrezzatura PRIMARY KEY (idAttrezzatura),
     CONSTRAINT fk_nome_attrezzatura FOREIGN KEY (nomeLab)
        REFERENCES azienda.laboratorio(Nome)
        ON DELETE SET NULL ON UPDATE CASCADE,
     CONSTRAINT fk_cup_attrezzatura FOREIGN KEY (CUP)
        REFERENCES azienda.progetto(CUP)
        ON DELETE CASCADE ON UPDATE CASCADE

    );
```

5.3.8 DIP_PROGETTO

```
CREATE TABLE azienda.DIP_PROGETTO(

     Matricola azienda.Matricola NOT NULL,
     Nome azienda.STRING NOT NULL,
     Cognome azienda.STRING NOT NULL,
     codFiscale azienda.CODFISCALE NOT NULL,
     Indirizzo VARCHAR(100),
     dataNascita DATE NOT NULL,
     dataAssunzione DATE NOT NULL,
     Costo azienda.EURO NOT NULL,
     Scadenza DATE NOT NULL,
     CUP azienda.CUP NOT NULL,

     CONSTRAINT pk_dip_progetto PRIMARY KEY (Matricola),
     CONSTRAINT fk_cup_dip_progetto FOREIGN KEY (CUP)
        REFERENCES azienda.progetto(CUP)
        ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT check_ordine_date_dp CHECK (dataNascita < dataAssunzione AND
        dataAssunzione <= Scadenza)

    );
```

5.4 Dizionario dei vincoli

Di seguito verranno riportati i vincoli basati sullo schema.
Successivamente, nella sezione relativa ai trigger, verranno analizzati i vincoli di integrità semantica.

Concetto	Vincolo	Descrizione
azienda.STRING (Dominio)	Tipo: VARCHAR(30)	**Vincolo di dominio** per gli attributi "Nome" e "Cognome" nella tabella DIP_INDETERMINATO, per gli attributi "Nome" e "Topic" nella tabella LABORATORIO, per l' attributo "nomeLab" nella tabella AFFERIRE, per l' attributo "Nome" nella tabella PROGETTO, per l' attributo "nomeLab" nella tabella LAVORARE, per gli attributi "Nome" e "Cognome" nella tabella DIP_PROGETTO.
azienda.EURO (Dominio)	Tipo: NUMERIC(20, 2)	**Vincolo di dominio** applicato agli attributi "Budget", "costoAttrezzature" e "costoContrattiProgetto" per la tabella PROGETTO, per l'attributo "Costo" per la tabella ATTREZZATURA e per l'attributo "Costo" per la tabella DIP_PROGETTO.
	POSITIVE_EURO	**Vincolo di dominio** che controlla che il costo di un acquisto o il budget di un progetto sia non negativo.
azienda.CUP (Dominio)	Tipo: VARCHAR(15)	**Vincolo di dominio** applicato all'attributo "CUP" nelle tabelle PROGETTO, LAVORARE, ATTREZZATURA e DIP_PROGETTO.
	DOM_CUP_CHECK_LENGTH	**Vincolo di dominio** che controlla che il CUP di un progetto rispetti la lunghezza di 15 caratteri.
	DOM_CUP_CHECK_ALPHANUMERIC	**Vincolo di dominio** che controlla che il CUP di un progetto sia alfanumerico.
azienda.COD-FISCALE (Dominio)	Tipo: VARCHAR(16)	**Vincolo di dominio** applicato all'attributo "codFiscale" nelle tabelle DIP_INDETERMINATO e DIP_PROGETTO.
	DOM_CODFISCALE_CHECK_LENGTH	**Vincolo di dominio** che controlla che il codice fiscale di un dipendente rispetti la lunghezza di 16 caratteri.
	DOM_CODFISCALE_CHECK_ALPHANUMERIC	**Vincolo di dominio** che controlla che il codice fiscale di un dipendente sia alfanumerico.
azienda.TIPO_DIPENDENTE (Dominio)	Tipo: VARCHAR(6)	**Vincolo di dominio** applicato all'attributo "Tipo" nella tabella DIP_INDETERMINATO.
	DOM_TIPO_DIPENDENTE_CHECK_JMS	**Vincolo di dominio** che controlla che il tipo di un dipendente sia: • *Junior*, • *Middle*, • *Senior*.

Concetto	Vincolo	Descrizione
azienda.TIPO_SCATTO (Dominio)	Tipo: VARCHAR(20)	**Vincolo di dominio** applicato all'attributo "Tipo" nella tabella SCATTO_CARRIERA.
	DOM_TIPO_SCATTO_CHECK_MSD	**Vincolo di dominio** che controlla che il tipo di scatto di un dipendente sia: • *Middle*, • *Senior*, • *Promosso a dirigente*, • *Rimosso da dirigente*.
DIP_INDET-ERMINATO (Relazione)	pk_dip_indeterminato	**Vincolo di chiave primaria** per l'attributo "Matricola"
	check_ordine_date_di	**Vincolo di dominio**, riguardante gli attributi "dataNascita", "dataAssunzione" e "dataFine", dove si controlla che non sia possibile assumere un dipendente non ancora nato oppure licenziare un dipendente non ancora assunto
SCATTO_CARRIERA (Relazione)	pk_scatto_carriera	**Vincolo di chiave primaria** che comprende gli attributi "Matricola", "Tipo" e "Data"
	fk_matricola_scatto_carriera	**Vincolo di chiave esterna** dell'attributo "Matricola", il quale fa riferimento all'attributo *chiave primaria* "Matricola" della relazione DIP_INDETERMINATO. Infatti, ogni scatto di carriera corrisponderà ad uno specifico dipendente a tempo indeterminato. Se si elimina o si aggiorna la matricola di un dipendente a tempo indeterminato, verranno cancellati o aggiornati anche tutti gli scatti ad esso relativi.
LABORATO-RIO (Relazione)	pk_laboratorio	**Vincolo di chiave primaria** per l'attributo "Nome"
	fk_responsabile_scientifico_laboratorio	**Vincolo di chiave esterna** dell'attributo "Responsabile_scientifico", il quale fa riferimento all'attributo *chiave primaria* "Matricola" della relazione DIP_INDETERMINATO. Infatti, ogni responsabile scientifico di un laboratorio corrisponde ad un dipendente a tempo indeterminato, ed ogni laboratorio non ne può essere sprovvisto. Se si intende eliminare un dipendente che è responsabile scientifico di un laboratorio, l'azione verrà impedita. Se si aggiorna la sua matricola, verrà aggiornata anche ogni sua occorrenza tra i responsabili scientifici.
	check_positive_nAfferenti	**Vincolo di dominio**, riguardante l'attributo "nAfferenti", dove si controlla che quest'ultimo non possa essere negativo. In particolare, deve essere strettamente maggiore di 0, poiché ogni laboratorio ha come minimo un'afferenza: quella del responsabile scientifico.

Concetto	Vincolo	Descrizione
AFFERIRE (Relazione)	pk_afferire	**Vincolo di chiave primaria** per gli attributi "Matricola" e "nomeLab".
	fk_matricola_afferire	**Vincolo di chiave esterna** dell'attributo "Matricola", il quale fa riferimento all'attributo *chiave primaria* "Matricola" della relazione "DIP_INDETERMINATO". Se si intende eliminare un dipendente a tempo indeterminato, verranno eliminate tutte le sue afferenze. Se, invece, si intende aggiornarne la matricola, verranno modificate tutte le sue occorrenze con la nuova.
	fk_nome_afferire	**Vincolo di chiave esterna** dell'attributo "nomeLab", il quale fa riferimento all'attributo *chiave primaria* "Nome" della relazione "LABORATORIO". Se si intende eliminare un laboratorio, verranno eliminate tutte le afferenze a quel laboratorio. Se, invece, si intende aggiornarne il nome del laboratorio, verranno modificate tutte le sue occorrenze.
PROGETTO (Relazione)	pk_progetto	**Vincolo di chiave primaria** per l'attributo "CUP".
	fk_referente_scientifico_progetto	**Vincolo di chiave esterna** dell'attributo "Referente_scientifico", il quale fa riferimento all'attributo *chiave primaria* "Matricola" della relazione "DIP_INDETERMINATO". Se si intende eliminare un dipendente che è un referente scientifico di un progetto, l'azione verrà impedita. Se, invece, si intende aggiornare la matricola, verranno modificate tutte le sue occorrenze.
	fk_responsabile_progetto	**Vincolo di chiave esterna** dell'attributo "Responsabile", il quale fa riferimento all'attributo *chiave primaria* "Matricola" della relazione "DIP_INDETERMINATO". Se si intende eliminare un dipendente che è un responsabile di un progetto, l'azione verrà impedita. Se, invece, si intende aggiornare la matricola, verranno modificate tutte le sue occorrenze.
	nome_progetto_unico	**Vincolo di unicità** assicura che il valore della colonna "Nome" in ogni riga della tabella sia unico.
	check_positive_budget	**Vincolo di dominio** riguardante l'attributo "Budget", dove si controlla che quest'ultimo non possa essere non positivo (o 0 oppure negativo).
	check_date_fine_inizio	**Vincolo di dominio** riguardante gli attributi "dataInizio" e "dataFine", dove si controlla che la data di inizio di un progetto non possa essere successiva alla data di fine.

Concetto	Vincolo	Descrizione
LAVORARE (Relazione)	pk_lavorare	**Vincolo di chiave primaria** che comprende gli attributi "CUP" e "nomeLab".
	fk_cup_lavorare	**Vincolo di chiave esterna** dell'attributo "CUP", il quale fa riferimento all'attributo *chiave primaria* "CUP" della relazione "PROGETTO". Se si intende eliminare un progetto, allora verranno eliminati anche tutti i rapporti di lavoro dei laboratori per quel progetto. Se, invece, si intende aggiornare il CUP, verranno modificate tutte le sue occorrenze.
	fk_nome_lavorare	**Vincolo di chiave esterna** dell'attributo "nomeLab", il quale fa riferimento all'attributo *chiave primaria* "Nome" della relazione "LABORATORIO". Se si intende eliminare un laboratorio, allora verranno eliminati anche tutti i rapporti di lavoro del laboratori per i progetti a lui corrispondenti. Se, invece, si intende modificare il nome del laboratorio, verranno modificate anche tutte le sue occorrenze.
ATTREZ-ZATURA (Relazione)	pk_attrezzatura	**Vincolo di primaria** per l'attributo "idAttrezzatura".
	fk_nome_attrezzatura	**Vincolo di chiave esterna** dell'attributo "nomeLab", il quale fa riferimento all'attributo *chiave primaria* "Nome" della relazione "LABORATORIO". Se si intende eliminare un laboratorio, allora la chiave esterna verrà modificata a NULL, poichè si vuole lasciare intatto lo storico degli acquisti fatti per un progetto. Se, invece, si intende modificare il nome del laboratorio che le contiene, verranno modificate tutte le sue occorrenze.
	fk_cup_attrezzatura	**Vincolo di chiave esterna** dell'attributo "CUP", il quale fa riferimento all'attributo *chiave primaria* "CUP" della relazione "PROGETTO". Se si intende eliminare un progetto, allora verranno eliminate tutte le attrezzature acquistate per quel progetto. Se, invece, si intende aggiornare il CUP, verranno modificate tutte le sue occorrenze.
DIP_PROGET-TO (Relazione)	pk_dip_progetto	**Vincolo di chiave primaria** per l'attributo "Matricola".
	fk_cup_dip_progetto	**Vincolo di chiave esterna** dell'attributo "CUP", il quale fa riferimento all'attributo *chiave primaria* "CUP" della relazione "PROGETTO". Se si intende eliminare un progetto, allora verranno eliminati anche tutti i contratti a progetto acquistati da esso. Se, invece, si intende aggiornare il CUP, verranno modificate tutte le sue occorrenze.
	check_ordine_date_dp	**Vincolo di dominio** riguardante gli attributi "dataNascita", "dataAssunzione" e "Scadenza", dove si impedisce di inserire un dipendente assunto prima della data di nascita o con data di scadenza del contratto antecedente alla data di assunzione.

5.5 Creazione viste

5.5.1 FondiRimanentiProgetto

La view "FondiRimanentiProgetto" consente di visualizzare i fondi rimanenti per ogni progetto dell'azienda, nonché la suddivisione di questi fondi tra attrezzature e contratti, così da poter visualizzare il budget rimanente per queste categorie di acquisti.

```
CREATE OR REPLACE VIEW azienda.FondiRimanentiProgetto AS

SELECT

     p.CUP,

     p.Nome,

     (p.Budget - p.costoAttrezzature - p.costoContrattiProgetto) AS
        Fondi_Rimanenti_Progetto,

     (p.Budget / 2) - (p.costoAttrezzature) AS Fondi_Rimanenti_Attrezzature,

     (p.Budget / 2) - (p.costoContrattiProgetto) AS Fondi_Rimanenti_Contratti

FROM

azienda.PROGETTO AS p;
```

5.5.2 dip_indeterminato_attuale

La view "dip_indeterminato_attuale" è progettata per mostrare solo le informazioni dei dipendenti a tempo indeterminato attualmente in servizio, basandosi sulla tabella "dip_indeterminato".

```
CREATE OR REPLACE VIEW azienda.dip_indeterminato_attuale AS

SELECT *

FROM azienda.dip_indeterminato AS DI

WHERE DI.DataFine IS NULL OR DI.DataFine > CURRENT_DATE;
```

5.5.3 scatto_carriera_attuale

La view "scatto_carriera_attuale" permette di visionare gli scatti di carriera dei dipendenti attualmente assunti dall'azienda.

```
CREATE OR REPLACE VIEW azienda.scatto_carriera_attuale AS

SELECT *

FROM azienda.scatto_carriera AS SC

WHERE SC.matricola IN (

     SELECT DI.matricola

     FROM azienda.dip_indeterminato AS DI

     WHERE DI.DataFine IS NULL OR DI.DataFine > CURRENT_DATE;

    );
```

5.5.4 progetto_attuale

La view "progetto_attuale" rappresenta una tabella che mostra i progetti dell'azienda non ancora terminati.

```
CREATE OR REPLACE VIEW azienda.progetto_attuale AS

SELECT *

FROM azienda.progetto AS P

WHERE P.DataFine >= now() OR P.DataFine IS NULL;
```

5.5.5 lavorare_attuale

La view "lavorare_attuale" restituisce le informazioni sui laboratori che lavorano attualmente ai progetti che sono ancora in corso o non hanno ancora una data di fine definita.

```
CREATE OR REPLACE VIEW azienda.lavorare_attuale AS

SELECT *

FROM azienda.lavorare AS LAV

WHERE LAV.CUP IN (

     SELECT CUP
     FROM azienda.progetto AS P
     P.DataFine >= now() OR P.DataFine IS NULL

    );
```

5.5.6 attrezzatura_attuale

La view "attrezzatura_attuale" mostrerà solo le attrezzature che sono attualmente assegnate ad un laboratorio, escludendo quelle che non sono assegnate a nessun laboratorio.

```
CREATE OR REPLACE VIEW azienda.attrezzatura_attuale AS

SELECT *

FROM azienda.attrezzatura

WHERE nomeLab IS NOT NULL;
```

5.5.7 dip_progetto_attuale

La view "dip_progetto_attuale" mostrerà solo i dipendenti assunti con contratto a progetto che lavorano a progetti attivi e il cui contratto è ancora in corso di validità.

```
CREATE OR REPLACE VIEW azienda.dip_progetto_attuale AS

SELECT DP.Matricola, DP.Nome, DP.Cognome, DP.codFiscale, DP.Indirizzo,
    DP.dataNascita, DP.dataAssunzione, DP.Costo, DP.Scadenza, DP.CUP

FROM azienda.dip_progetto AS DP JOIN azienda.progetto AS P ON DP.CUP = P.CUP

WHERE (P.DataFine >= now() OR P.DataFine IS NULL) AND DP.Scadenza >= now();
```

5.5.8 ProgettiLab

La view "ProgettiLab" mostrerà i progetti ai quali lavorano meno di 3 laboratori, a cui quindi è possibile assegnarne uno nuovo.

```
CREATE OR REPLACE VIEW azienda.Progettilab AS

SELECT CUP, COUNT(*) AS "nLab lavoranti"

FROM azienda.LAVORARE

GROUP BY CUP

HAVING COUNT(*) < 3;
```

5.6 Creazione funzioni

5.6.1 get_list_CUP_referente_scientifico

La funzione "get_list_CUP_referente_scientifico" restituisce una stringa contenente un elenco di CUP (Codice Unico di Progetto) ai quali un determinato referente scientifico (identificato dalla matricola presa in input) è assegnato.

```
CREATE OR REPLACE FUNCTION azienda.get_list_CUP_referente_scientifico(matricola IN
    azienda.matricola)

RETURNS TEXT

AS

$$

DECLARE

     progettiCoinvolti CURSOR FOR

          SELECT CUP

          FROM azienda.progetto

          WHERE Referente_Scientifico = matricola AND (dataFine IS NULL OR dataFine
             >= current_date);

     cup_progetto azienda.CUP;

     lista_referente_scientifico TEXT := '';

BEGIN

     OPEN progettiCoinvolti;

     LOOP

          FETCH progettiCoinvolti INTO cup_progetto;

          EXIT WHEN NOT FOUND;

          lista_referente_scientifico = CONCAT (lista_referente_scientifico,
             cup_progetto || ', ');

     END LOOP;

     CLOSE progettiCoinvolti;

     IF lista_referente_scientifico <> '' THEN

          lista_referente_scientifico = SUBSTR(lista_referente_scientifico, 1,
             LENGTH(lista_referente_scientifico)-2);

     END IF;

     RETURN lista_referente_scientifico;

END;

$$

LANGUAGE plpgsql;
```

5.6.2 get_list_CUP_responsabile_progetto

La funzione "get_list_CUP_responsabile_progetto" restituisce una stringa contenente un elenco di CUP (Codice Unico di Progetto) ai quali un determinato responsabile di un progetto (identificato dalla matricola presa in input) è assegnato.

```
CREATE OR REPLACE FUNCTION azienda.get_list_CUP_responsabile_progetto(matricola IN
    azienda.matricola)

RETURNS TEXT

AS

$$

DECLARE

     progettiCoinvolti CURSOR FOR
          SELECT CUP
          FROM azienda.progetto
          WHERE Responsabile = matricola AND (dataFine IS NULL OR dataFine >=
             current_date);
     cup_progetto azienda.CUP;
     lista_responsabile TEXT := '';

BEGIN

     OPEN progettiCoinvolti;
     LOOP
          FETCH progettiCoinvolti INTO cup_progetto;
          EXIT WHEN NOT FOUND;
          lista_responsabile = CONCAT (lista_responsabile, cup_progetto || ', ');
     END LOOP;
     CLOSE progettiCoinvolti;


     IF lista_responsabile <> '' THEN
          lista_responsabile = SUBSTR(lista_responsabile, 1,
             LENGTH(lista_responsabile)-2);
     END IF;
     RETURN lista_responsabile;

END;

$$

LANGUAGE plpgsql;
```

5.6.3 get_list_responsabile_laboratorio

La funzione "get_list_responsabile_laboratorio" ha lo scopo di recuperare la lista dei laboratori di cui la matricola inserita come input è il responsabile scientifico.

```
CREATE OR REPLACE FUNCTION azienda.get_list_responsabile_laboratorio(matricola IN
    azienda.matricola)

RETURNS TEXT

AS

$$

DECLARE

    laboratoriCoinvolti CURSOR FOR
        SELECT Nome
        FROM azienda.laboratorio
        WHERE Responsabile_Scientifico = matricola;
    lab azienda.laboratorio.nome%TYPE;
    lista_responsabile_scientifico TEXT := '';

BEGIN

    OPEN laboratoriCoinvolti;
    LOOP
        FETCH laboratoriCoinvolti INTO lab;
        EXIT WHEN NOT FOUND;
        lista_responsabile_scientifico = CONCAT (lista_responsabile_scientifico,
          lab || ', ');
    END LOOP;
    CLOSE laboratoriCoinvolti;
    IF lista_responsabile_scientifico <> '' THEN
        lista_responsabile_scientifico = SUBSTR(lista_responsabile_scientifico, 1,
          LENGTH(lista_responsabile_scientifico)-2);
    END IF;
    RETURN lista_responsabile_scientifico;

END;

$$

LANGUAGE plpgsql;
```

5.6.4 aggiorna_tipo

La funzione "aggiorna_tipo" aggiorna il tipo di un dipendente a tempo indeterminato in base alla differenza di anni tra la data di assunzione e la data di fine (se presente) o la data attuale.
Se la differenza di anni è inferiore a 3, il tipo del dipendente viene impostato a Junior; se è compresa tra 3 e 7 anni, il tipo viene impostato a Middle; se è superiore a 7 anni, il tipo viene impostato a Senior.

```
CREATE OR REPLACE FUNCTION azienda.aggiorna_tipo(

    mat IN azienda.dip_indeterminato.matricola%TYPE,

    tipo_mat IN azienda.dip_indeterminato.tipo%TYPE,

    dataAssunzione_mat IN azienda.dip_indeterminato.dataAssunzione%TYPE,

    dataFine_mat IN azienda.dip_indeterminato.dataFine%TYPE

   )

RETURNS azienda.dip_indeterminato.tipo%TYPE

AS

$$

DECLARE

    check_existance azienda.dip_indeterminato.matricola%TYPE;

    numero_anni_trascorsi INTEGER := DATE_PART('year', AGE(COALESCE(dataFine_mat,
       CURRENT_DATE), dataAssunzione_mat));

BEGIN

    SELECT Matricola INTO check_existance

    FROM azienda.dip_indeterminato

    WHERE Matricola = mat AND Tipo = tipo_mat AND dataAssunzione =
       dataAssunzione_mat AND dataFine IS NOT DISTINCT FROM dataFine_mat;

    IF NOT FOUND THEN

        IF tipo_mat IS NULL OR dataAssunzione_mat IS NULL THEN

            RAISE EXCEPTION 'Matricola % non esistente!', mat;

        ELSE

            RAISE EXCEPTION 'Matricola % di tipo % e assunto in data % non
              esistente!', mat, tipo_mat, dataAssunzione_mat;

        END IF;

    END IF;

    IF numero_anni_trascorsi < 3 THEN

        IF tipo_mat = 'Middle' OR tipo_mat = 'Senior' THEN

            RAISE NOTICE La matricola % è di tipo ''%'' anche se non ha trascorso 3
              anni in azienda!  Cambio il tipo in ''Junior''', mat, tipo_mat;

        END IF;

        tipo_mat = 'Junior';

    ELSIF 3 <= numero_anni_trascorsi AND numero_anni_trascorsi < 7 THEN

        IF tipo_mat = 'Junior' THEN
```

```
            RAISE NOTICE 'La matricola % è di tipo ''Junior'' ma ha trascorso più
              di 3 anni in azienda!  Cambio il tipo in ''Middle''', mat;
        ELSIF tipo_mat = 'Senior' THEN
            RAISE NOTICE 'La matricola % è di tipo ''Senior'' anche se non ha
              trascorso 7 anni in azienda!  Cambio il tipo in ''Middle''', mat;
        END IF;
        tipo_mat = 'Middle';
    ELSIF numero_anni_trascorsi >= 7 THEN
        IF tipo_mat <> 'Senior' THEN
            RAISE NOTICE 'La matricola % è di tipo ''%'' ma ha trascorso più di 7
              anni in azienda!  Cambio il tipo in ''Senior''', mat, tipo_mat;
        END IF;
        tipo_mat = 'Senior';
    END IF;
    RETURN tipo_mat;
END;
$$
LANGUAGE plpgsql;
```

5.6.5 get_list_lab_afferenza_matricola

La funzione "get_list_lab_afferenza_matricola" ha lo scopo di restituire i nomi di tutti i laboratori a cui afferisce una determinata matricola.

```
CREATE OR REPLACE FUNCTION azienda.get_list_lab_afferenza_matricola(dip_matricola IN
    azienda.Matricola)

RETURNS TEXT

AS

$$

DECLARE

     var RECORD;
     lista_nomiLab text := '';

BEGIN

     FOR var IN
          SELECT nomeLab
          FROM azienda.AFFERIRE
          WHERE Matricola = dip_matricola
     LOOP
          IF LENGTH(lista_nomiLab) = 0 THEN
              lista_nomiLab := var.nomeLab;
          ELSE
              lista_nomiLab := CONCAT(lista_nomiLab, ', ', var.nomeLab);
          END IF;
     END LOOP;
     RETURN lista_nomiLab;

END;

$$

LANGUAGE plpgsql;
```

5.7 Creazione procedure

5.7.1 sostituisci_referente_scientifico

La procedura "sostituisci_referente_scientifico" sostituisce il referente scientifico di uno o più progetti, identificati dal CUP, con una nuova matricola fornita in input.

```
CREATE OR REPLACE PROCEDURE azienda.sostituisci_referente_scientifico(listaCUP IN
    TEXT, matricola IN azienda.matricola)

AS

$$

DECLARE

     copyListaCUP TEXT := listaCUP;
     singleCUP azienda.CUP;
     separatoreIndex INT;;

BEGIN

     IF listaCUP = '' THEN
         RAISE EXCEPTION 'Non ci sono CUP in input';
     END IF;
     separatoreIndex := STRPOS(listaCUP, ', ');

     WHILE separatoreIndex <> 0
     LOOP
         singleCUP := SUBSTR(listaCUP, 1, separatoreIndex-1);
         listaCUP := SUBSTR(listaCUP, separatoreIndex+2, LENGTH(listaCUP));

         UPDATE azienda.progetto
         SET Referente_Scientifico = matricola
         WHERE CUP = singleCUP;

         separatoreIndex := STRPOS(listaCUP, ', ');
     END LOOP;
     UPDATE azienda.progetto
     SET Referente_Scientifico = matricola
     WHERE CUP = listaCUP;
     RAISE NOTICE 'Ho sostituito il Referente scientifico dei progetti:  % con la
        matricola ''%''', copyListaCUP, matricola;

END;

$$

LANGUAGE plpgsql;
```

5.7.2 sostituisci_responsabile_progetto

La procedura "sostituisci_responsabile_progetto" sostituisce il Responsabile di uno o più progetti (identificati dalla loro lista di CUP) con un altro dipendente, il cui identificativo di matricola è specificato come parametro di input.

```
CREATE OR REPLACE PROCEDURE azienda.sostituisci_responsabile_progetto(listaCUP IN
    TEXT, matricola IN azienda.matricola)

AS

$$

DECLARE

     copyListaCUP TEXT := listaCUP;
     singleCUP azienda.CUP;
     separatoreIndex INT;

BEGIN

     IF listaCUP = '' THEN
         RAISE EXCEPTION 'Non ci sono CUP in input';
     END IF;
     separatoreIndex := STRPOS(listaCUP, ', ');

     WHILE separatoreIndex <> 0
     LOOP
         singleCUP := SUBSTR(listaCUP, 1, separatoreIndex-1);
         listaCUP := SUBSTR(listaCUP, separatoreIndex+2, LENGTH(listaCUP));

         UPDATE azienda.progetto
         SET Responsabile = matricola
         WHERE CUP = singleCUP;

         separatoreIndex := STRPOS(listaCUP, ', ');
     END LOOP;
     UPDATE azienda.progetto
     SET Responsabile = matricola
     WHERE CUP = listaCUP;
     RAISE NOTICE 'Ho sostituito il Responsabile dei progetti:  % con la matricola
        ''%''', copyListaCUP, matricola;

END;

$$

LANGUAGE plpgsql;
```

5.7.3 sostituisci_responsabile_laboratorio

La procedura "sostituisci_responsabile_laboratorio" si occupa di sostituire il Responsabile Scientifico di uno o più laboratori con un altro dipendente, il cui identificativo di matricola è specificato come parametro di input.

```
CREATE OR REPLACE PROCEDURE
    azienda.sostituisci_responsabile_laboratorio(listaLaboratori IN TEXT, matricola
    IN azienda.matricola)

AS

$$

DECLARE

     copylistaLaboratori TEXT := listaLaboratori;
     lab azienda.laboratorio.nome%TYPE;
     separatoreIndex INT;

BEGIN

     IF listaLaboratori = '' THEN
         RAISE EXCEPTION 'Non ci sono CUP in input';
     END IF;
     separatoreIndex := STRPOS(listaLaboratori, ', ');


     WHILE separatoreIndex <> 0
     LOOP
         lab := SUBSTR(listaLaboratori, 1, separatoreIndex-1);
         listaLaboratori := SUBSTR(listaLaboratori, separatoreIndex+2,
           LENGTH(listaLaboratori));

         UPDATE azienda.laboratorio
         SET Responsabile_Scientifico = matricola
         WHERE nome = lab;

         separatoreIndex := STRPOS(listaLaboratori, ', ');
     END LOOP;
     UPDATE azienda.laboratorio
     SET Responsabile_Scientifico = matricola
     WHERE nome = listaLaboratori;
     RAISE NOTICE 'Ho sostituito il Responsabile scientifico dei laboratori:  % con
        la matricola ''%''', copylistaLaboratori, matricola;

END;

$$

LANGUAGE plpgsql;
```

5.7.4 check_scatto

La procedura "check_scatto" ha lo scopo di registrare gli scatti di carriera di un dipendente indeterminato dell'azienda. Dati in input la matricola del dipendente, il tipo da verificare e la data di assunzione (necessaria per la corretta registrazione della data dello scatto), la procedura verifica che ci sia coerenza tra il "Tipo" dichiarato in input e gli scatti presenti nella tabella "SCATTO_CARRIERA". Nel caso in cui non ci sia coerenza, vengono aggiunti gli scatti mancanti. Gli scatti di carriera vengono registrati in date specifiche (dopo 3 anni dalla data di assunzione per lo scatto a "Middle" e dopo 7 anni per lo scatto a "Senior").

```
CREATE OR REPLACE PROCEDURE azienda.check_scatto(

    mat IN azienda.dip_indeterminato.matricola%TYPE,
    tipo_mat IN azienda.dip_indeterminato.tipo%TYPE,
    dataAssunzione_mat IN azienda.dip_indeterminato.dataAssunzione%TYPE

   )

AS

$$

DECLARE

    check_existance azienda.dip_indeterminato.matricola%TYPE;
    cursore CURSOR (test_tipo TEXT) FOR
        SELECT Data
        FROM azienda.SCATTO_CARRIERA
        WHERE Matricola = mat AND tipo = test_tipo;
    data_scatto azienda.scatto_carriera.data%TYPE;
    scattiPresenti TEXT := 'Scatti giá registrati:';
    scattiAggiunti TEXT := 'Nuovi scatti registrati:';

BEGIN

    SELECT Matricola INTO check_existance
    FROM azienda.dip_indeterminato
    WHERE Matricola = mat AND UPPER(Tipo) = UPPER(tipo_mat) AND dataAssunzione =
       dataAssunzione_mat;
    IF NOT FOUND THEN
        IF tipo_mat IS NULL OR dataAssunzione_mat IS NULL THEN
            RAISE EXCEPTION 'Matricola % non esistente!', mat;
        ELSE
            RAISE EXCEPTION 'Matricola % di tipo % e assunto in data % non
              esistente!', mat, tipo_mat, dataAssunzione_mat;
        END IF;
    END IF;

    IF UPPER(tipo_mat) = 'JUNIOR' THEN tipo_mat = 'Junior';
    ELSIF UPPER(tipo_mat) = 'MIDDLE' THEN tipo_mat = 'Middle';
```

```
ELSIF UPPER(tipo_mat) = 'SENIOR' THEN tipo_mat = 'Senior';
ELSE RAISE EXCEPTION 'Tipo non accettato!';
END IF;

IF tipo_mat = 'Junior' THEN
    RAISE NOTICE 'La matricola % è di tipo Junior!  Nessuno scatto
      registrato', mat;
ELSE
    OPEN cursore('Middle');
    FETCH cursore INTO data_scatto;
    IF NOT FOUND THEN
        scattiAggiunti := CONCAT (scattiAggiunti, ' "Junior" a "Middle"');
        INSERT INTO azienda.SCATTO_CARRIERA(Matricola, Tipo, Data) VALUES
        (mat, 'Middle', dataAssunzione_mat + interval '3 years');
    ELSE
        scattiPresenti := CONCAT (scattiPresenti, ' "Junior" a "Middle"');
    END IF;
    CLOSE cursore;

    IF tipo_mat = 'Senior' THEN
        OPEN cursore('Senior');
        FETCH cursore INTO data_scatto;
        IF NOT FOUND THEN
            IF scattiAggiunti <> 'Nuovi scatti registrati:'  THEN
              scattiAggiunti := CONCAT (scattiAggiunti, ' e');
            END IF;
            scattiAggiunti := CONCAT (scattiAggiunti, ' "Middle" a "Senior"');
            INSERT INTO azienda.SCATTO_CARRIERA(Matricola, Tipo, Data) VALUES
            (mat, 'Senior', dataAssunzione_mat + interval '7 years');
        ELSE
            IF scattiPresenti <> 'Scatti già registrati:'  THEN
              scattiPresenti := CONCAT (scattiPresenti, ' e');
            END IF;
            scattiPresenti := CONCAT (scattiPresenti, ' "Middle" a "Senior"');
        END IF;
        CLOSE cursore;
    END IF;

    IF scattiPresenti = 'Scatti già registrati:'  THEN
        scattiPresenti := CONCAT(scattiPresenti, ' Nessuno');
    END IF;

    IF scattiAggiunti = 'Nuovi scatti registrati:'  THEN
        scattiAggiunti := CONCAT(scattiAggiunti, ' Nessuno');
```

```
        END IF;

        RAISE NOTICE E'Matricola %, tipo %:\n% \n%', mat, tipo_mat,
          scattiPresenti, scattiAggiunti;
    END IF;
END;
$$
LANGUAGE plpgsql;
```

5.7.5 aggiorna_tipo_listaDipendenti

La procedura "aggiorna_tipo_listaDipendenti" ha lo scopo di aggiornare l'anzianità di una lista di dipendenti passati per input. In particolare, verifica l'anzianità di ogni dipendente sia aggiornata rispetto alla data odierna (o alla data di licenziamento, se presente). Se vi sono dipendenti da aggiornare, li aggiorna. Assumiamo che tale procedura venga invocata periodicamente per aggiornare i dipendenti che con il tempo hanno fatto uno scatto di carriera.

```
CREATE OR REPLACE PROCEDURE azienda.aggiorna_tipo_listaDipendenti(listaMatricole IN
    TEXT)
AS
$$
DECLARE
    copyListaMatricole TEXT := listaMatricole;
    dati_matricola RECORD;
    tipo_effettivo azienda.dip_indeterminato.tipo%TYPE;
    mat azienda.dip_indeterminato.matricola%TYPE;
    separatoreIndex INT;
BEGIN
    IF listaMatricole = '' THEN
        RAISE EXCEPTION 'Non ci sono matricole in input';
    END IF;
    separatoreIndex := STRPOS(listaMatricole, ', ');

    WHILE separatoreIndex <> 0
    LOOP
```

```
        mat := SUBSTR(listaMatricole, 1, separatoreIndex-1);
        listaMatricole := SUBSTR(listaMatricole, separatoreIndex+2,
          LENGTH(listaMatricole));

        SELECT tipo, dataAssunzione, dataFine INTO dati_matricola
        FROM azienda.DIP_INDETERMINATO
        WHERE Matricola = mat;

        IF NOT FOUND THEN
            RAISE EXCEPTION 'La matricola % non esiste', mat;
        END IF;

        tipo_effettivo = azienda.aggiorna_tipo(mat, dati_matricola.tipo,
          dati_matricola.dataAssunzione, dati_matricola.dataFine);
        IF tipo_effettivo <> dati_matricola.tipo THEN
            UPDATE azienda.DIP_INDETERMINATO
            SET Tipo = tipo_effettivo
            WHERE Matricola = mat;
        END IF;

        separatoreIndex := STRPOS(listaMatricole, ', ');
    END LOOP;
    SELECT tipo, dataAssunzione, dataFine INTO dati_matricola
    FROM azienda.DIP_INDETERMINATO
    WHERE Matricola = listaMatricole;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'La matricola % non esiste', listaMatricole;
    END IF;

    tipo_effettivo = azienda.aggiorna_tipo(listaMatricole, dati_matricola.tipo,
       dati_matricola.dataAssunzione, dati_matricola.dataFine);
    IF tipo_effettivo <> dati_matricola.tipo THEN
        UPDATE azienda.DIP_INDETERMINATO
        SET Tipo = tipo_effettivo
        WHERE Matricola = listaMatricole;
    END IF;

    RAISE NOTICE 'Ho aggiornato, dove possibile, l''anzianità per le matricole:
       %', copyListaMatricole;

END;

$$

LANGUAGE plpgsql;
```

5.7.6 check_afferenza_per_dataFine

La procedura "check_afferenza_per_dataFine" ha lo scopo di eliminare tutte le afferenze ai laboratori per i dipendenti a tempo indeterminato della tabella "azienda.DIP_INDETERMINATO" che hanno una data di fine contratto minore o uguale alla data corrente e che risultano essere ancora afferiti ad almeno un laboratorio nella tabella "azienda.AFFERIRE". Assumiamo che tale procedura venga invocata periodicamente.

```
CREATE OR REPLACE PROCEDURE azienda.check_afferenza_per_dataFine()

AS

$$

DECLARE

     lista_laboratori_afferiti text;
     var RECORD;

BEGIN

     FOR var IN
          SELECT DI.Matricola
          FROM azienda.DIP_INDETERMINATO AS DI
          WHERE DI.dataFine <= DATE(NOW() AND EXISTS
              (SELECT *
              FROM azienda.AFFERIRE AS A
              WHERE A.matricola = DI.Matricola))
     LOOP
          lista_laboratori_afferiti :=
            azienda.get_list_lab_afferenza_matricola(var.Matricola);

          IF lista_laboratori_afferiti = '' THEN
              RAISE NOTICE E'Il dipendente a tempo indeterminato % non afferisce ad
                alcun laboratorio.\n Pertanto non è possibile rimuovere alcuna
                afferenza per quest''ultimo!', var.Matricola;
          ELSE
              DELETE FROM azienda.AFFERIRE
              WHERE Matricola = var.Matricola;
          RAISE NOTICE E'Sono state eliminate le afferenze ai laboratori "%" del
            dipendente a tempo indeterminato "%"', lista_laboratori_afferiti,
            var.Matricola;
          END IF;
     END LOOP;

END;

$$

LANGUAGE plpgsql;
```

5.8 Creazione trigger

Di seguito sono riportati tutti i vincoli di integrità semantica, accompagnati da un'apposita descrizione

5.8.1 TRIGGERS PER azienda.DIP_INDETERMINATO

nice_looking_domain_di

Il trigger "nice_looking_domain_di" rende il campo "Tipo" di "azienda.DIP_INDETERMINATO" case insensitive. Il campo "Tipo" avrà sempre la prima lettera maiuscola e le restanti minuscole.

```
CREATE OR REPLACE FUNCTION azienda.fn_nice_looking_domain_di()

RETURNS trigger

AS

$$

BEGIN

     NEW.tipo = CONCAT(UPPER(SUBSTR(NEW.tipo, 1, 1)), LOWER(SUBSTR(NEW.Tipo, 2,
        LENGTH(NEW.Tipo))));

     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER a_tr_nice_looking_domain_di

BEFORE INSERT OR UPDATE OF tipo ON azienda.DIP_INDETERMINATO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_nice_looking_domain_di();
```

blocco_modifica_dirigente

Il trigger "blocco_modifica_dirigente" impedisce la modifica manuale dello stato dirigenziale dei dipendenti nella tabella "azienda.DIP_INDETERMINATO".
Viene distinto il caso dell'inserimento da quello dell'aggiornamento del campo "Dirigente". Nel caso dell'inserimento viene impostato automaticamente a FALSE, nel caso dell'update viene ignorata la modifica effettuata. Questo perché la dirigenza viene aggiornata sulla base dello scatto inserito in "azienda.SCATTO_CARRIERA".

```
CREATE OR REPLACE FUNCTION azienda.fn_blocco_modifica_dirigente()

RETURNS trigger

AS

$$

BEGIN

     IF OLD.Matricola IS NULL THEN
         IF (NEW.Dirigente <> FALSE) THEN
             RAISE NOTICE E'Non puoi inserire manualmente lo stato dirigenziale del
               dipendete %.  E'' necessario inserire l''apposito scatto in
               "azienda.scatto_carriera"', NEW.Matricola;
             NEW.Dirigente = FALSE;
         END IF;
     ELSE
         IF (NEW.Dirigente <> OLD.Dirigente) THEN
             RAISE NOTICE E'Non puoi inserire manualmente lo stato dirigenziale del
               dipendete %.  E'' necessario inserire l''apposito scatto in
               "azienda.scatto_carriera"', NEW.Matricola;
             NEW.Dirigente = OLD.Dirigente;
           END IF;
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_blocco_modifica_dirigente

BEFORE INSERT OR UPDATE OF Dirigente ON azienda.DIP_INDETERMINATO

FOR EACH ROW

WHEN (pg_trigger_depth() < 1)

EXECUTE FUNCTION azienda.fn_blocco_modifica_dirigente();
```

verifica_tipo_dipInd

Il trigger "verifica_tipo_dipInd" controlla il tipo di un dipendente in base al numero di anni trascorsi dall'assunzione. In particolare, la funzione controlla se il tipo di un dipendente rispetta i requisiti di anzianità per il ruolo assegnato (Junior, Middle o Senior).

```
CREATE OR REPLACE FUNCTION azienda.fn_verifica_tipo_dipInd()

RETURNS trigger

AS

$$

DECLARE

    numero_anni_trascorsi INTEGER := DATE_PART('year', AGE(COALESCE(NEW.dataFine,
       CURRENT_DATE), NEW.dataAssunzione));

BEGIN

    IF numero_anni_trascorsi < 3 THEN
        IF NEW.Tipo = 'Middle' OR NEW.Tipo = 'Senior' THEN
            RAISE EXCEPTION 'La matricola % è di tipo "%" anche se non ha trascorso
              3 anni in azienda!', NEW.matricola, NEW.Tipo;
        END IF;
    ELSIF 3 <= numero_anni_trascorsi AND numero_anni_trascorsi < 7 THEN
        IF NEW.Tipo = 'Junior' THEN
            RAISE EXCEPTION 'La matricola % è di tipo "Junior" ma ha trascorso più
              di 3 anni in azienda!', NEW.matricola;
        ELSIF NEW.Tipo = 'Senior' THEN
            RAISE EXCEPTION 'La matricola % è di tipo "Senior" anche se non ha
              trascorso 7 anni in azienda!', NEW.matricola;
        END IF;
    ELSIF numero_anni_trascorsi >= 7 THEN
        IF NEW.Tipo <> 'Senior' THEN
            RAISE EXCEPTION 'La matricola % è di tipo "%" ma ha trascorso più di 7
              anni in azienda!', NEW.matricola, NEW.Tipo;
        END IF;
    END IF;
    RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER b_tr_verifica_tipo_dipInd

BEFORE INSERT OR UPDATE ON azienda.DIP_INDETERMINATO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_verifica_tipo_dipInd();
```

aggiorna_scatti_tipo

Il trigger "aggiorna_scatti_tipo" aggiorna automaticamente gli scatti di carriera dei dipendenti di un'azienda nel caso in cui vengano inseriti o aggiornati i loro dati nel database. Nel caso in cui il dipendente sia stato promosso o declassato (caso possibile in seguito all'aggiornamento del periodo di attività), la funzione elimina eventuali scatti di carriera superflui e aggiunge quelli mancanti.

```
CREATE OR REPLACE FUNCTION azienda.fn_aggiorna_scatti_tipo()

RETURNS trigger

AS

$$

BEGIN

     IF OLD.Matricola IS NULL THEN
         CALL azienda.check_scatto(NEW.Matricola, NEW.Tipo, NEW.dataAssunzione);
     ELSE;
         IF NEW.Tipo = 'Junior' AND (OLD.Tipo = 'Middle' OR OLD.Tipo = 'Senior')
            THEN
             DELETE FROM azienda.SCATTO_CARRIERA
             WHERE Matricola = NEW.Matricola AND (Tipo = 'Middle' OR Tipo =
               'Senior');
         ELSIF NEW.Tipo = 'Middle' AND (OLD.Tipo = 'Senior') THEN
             DELETE FROM azienda.SCATTO_CARRIERA
             WHERE Matricola = NEW.Matricola AND Tipo = 'Senior';
         CALL azienda.check_scatto(NEW.Matricola, NEW.Tipo, NEW.dataAssunzione);
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER b_tr_aggiorna_scatti_tipo

AFTER INSERT OR UPDATE ON azienda.DIP_INDETERMINATO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_aggiorna_scatti_tipo();
```

assunzione_coerente

Il trigger "assunzione_coerente" verifica se l'assunzione di un dipendente risulta coerente rispetto ad altri contratti di lavoro esistenti. In particolare, controlla se ci sono conflitti tra i periodi di validità dei contratti (inizio e fine), ovvero se ci sono sovrapposizioni o periodi di tempo in cui un dipendente ha due o più contratti contemporaneamente.

```
CREATE OR REPLACE FUNCTION azienda.fn_assunzione_coerente()

RETURNS trigger

AS

$$

DECLARE

    cursore CURSOR FOR
        SELECT DI.Matricola, DI.codFiscale, DI.dataAssunzione, DI.dataFine
        FROM azienda.DIP_INDETERMINATO DI
        WHERE DI.codFiscale = NEW.codFiscale AND DI.matricola <> NEW.Matricola
        ORDER BY DI.dataAssunzione ASC;
    dip_ind RECORD;
    lista_contratti TEXT := '';

BEGIN

    OPEN cursore;
    LOOP


        FETCH cursore INTO dip_ind;
        EXIT WHEN NOT FOUND;
        IF (((dip_ind.dataFine IS NULL) AND (dip_ind.Matricola <> NEW.Matricola))
          OR
        (dip_ind.dataAssunzione <= NEW.dataAssunzione AND NEW.dataAssunzione <
          dip_ind.dataFine) OR
        (dip_ind.dataAssunzione < NEW.dataFine AND NEW.dataFine <=
          dip_ind.dataFine) OR
        NEW.dataAssunzione < dip_ind.dataAssunzione AND NEW.dataFine >
          dip_ind.dataFine) THEN
           lista_contratti := CONCAT(lista_contratti, dip_ind.Matricola || ', ');
        END IF;
    END LOOP;
    CLOSE cursore;


    IF lista_contratti <> '' THEN
        lista_contratti := SUBSTR(lista_contratti, 1, LENGTH(lista_contratti)-2);
        RAISE EXCEPTION 'Non è stato possibile stipulare il contratto con
          matricola % per %.
          La persona ha ancora un contratto in corso o vi è un conflitto tra la
          data di assunzione (o di fine) e i seguenti contratti:  %',
          NEW.Matricola, NEW.codFiscale, lista_contratti;
```

```
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER c_tr_assunzione_coerente
BEFORE INSERT OR UPDATE OF codFiscale, dataAssunzione, dataFine
ON azienda.DIP_INDETERMINATO
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_assunzione_coerente();
```

delete_dipInd_afferenze

Il trigger "delete_dipInd_afferenze" verifica se il valore della colonna "dataFine" è stato modificato, e se la nuova data è antecedente a quella corrente, elimina tutte le tuple correlate all'impiegato nella tabella "AFFERIRE". In caso contrario, la funzione mantiene le afferenze sino alla data specificata.

```
CREATE OR REPLACE FUNCTION azienda.fn_delete_dipInd_afferenze()

RETURNS trigger

AS

$$

BEGIN

     IF NEW.dataFine <= DATE(NOW()) THEN
          DELETE FROM azienda.AFFERIRE
          WHERE Matricola = OLD.Matricola;
          RAISE NOTICE 'Eliminate tutte le afferenze della matricola % non più
             attiva!', OLD.Matricola;
     ELSE
          RAISE NOTICE 'Siccome la dataFine inserita % è successiva rispetto la data
             corrente %, verranno mantenute tutte le afferenze semplici ai
             laboratori.',NEW.dataFine, DATE(NOW());
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_delete_dipInd_afferenze

AFTER UPDATE OF dataFine ON azienda.DIP_INDETERMINATO

FOR EACH ROW

WHEN (OLD.dataFine IS DISTINCT FROM NEW.DataFine AND NEW.dataFine IS NOT NULL)

EXECUTE FUNCTION azienda.fn_delete_dipInd_afferenze();
```

licenzia_dipInd_con_incarichi

Il trigger "licenzia_dipInd_con_incarichi" ha l'obiettivo di impedire il licenziamento (o la modifica del tipo o della dirigenza) di un dipendente dell'azienda che ricopre il ruolo di Referente scientifico o di Responsabile di progetti attivi o che ricopre il ruolo di Responsabile scientifico di un laboratorio.

```
CREATE OR REPLACE FUNCTION azienda.fn_licenzia_dipInd_con_incarichi()

RETURNS trigger

AS

$$

DECLARE

    lista_responsabile TEXT :=
       azienda.get_list_CUP_responsabile_progetto(NEW.matricola);
    lista_referente_scientifico TEXT :=
       azienda.get_list_CUP_referente_scientifico(NEW.matricola);
    lista_responsabile_scientifico TEXT :=
       azienda.get_list_responsabile_laboratorio(NEW.matricola);

BEGIN

    IF (OLD.Tipo <> NEW.Tipo) OR (NEW.DataFine IS NOT NULL) THEN
        IF lista_referente_scientifico <> '' THEN
            RAISE EXCEPTION E'Impossibile licenziare il dipendente (o modificarne
              il tipo) con matricola % perchè è ancora Referente scientifico in
              alcuni progetti attivi.  \n
              Sostituisci prima il Referente scientifico (è possibile farlo
              tramite la procedura "azienda.sostituisci_referente_scientifico
              (azienda.get_list_CUP_referente_scientifico (<vecchiaMatricola>),
              <nuovaMatricola> )", che sostituirà tutte le vecchie occorrenze del
              referente scientifico con la nuova matricola) e poi potrai procedere
              con il licenziamento.\n I progetti in questione sono:  %',
              NEW.Matricola, lista_referente_scientifico;
        END IF;
        IF lista_responsabile_scientifico <> '' THEN
            RAISE EXCEPTION E'Impossibile licenziare il dipendente (o modificarne
              il tipo) con matricola % perchè è ancora Responsabile scientifico in
              alcuni laboratori.\n
              Sostituisci prima il Responsabile scientifico (è possibile farlo
              tramite la procedura "azienda.sostituisci_responsabile_laboratorio
              (azienda.get_list_responsabile_laboratorio (<vecchiaMatricola>),
              <nuovaMatricola>)", che sostituirà tutte le vecchie occorrenze del
              responsabile scientifico con la nuova matricola) e poi potrai
              procedere con il licenziamento.\nI laboratori in questione sono:
              %', NEW.Matricola, lista_responsabile_scientifico;
        END IF;
    END IF;


    IF (OLD.Dirigente <> NEW.Dirigente) OR (NEW.DataFine IS NOT NULL) THEN
```

```
        IF lista_responsabile <> '' THEN
            RAISE EXCEPTION E'Impossibile licenziare il dipendente (o modificarne
              la dirigenza) con matricola % perchè è ancora Responsabile in alcuni
              progetti attivi.\n
              Sostituisci prima il Responsabile (è possibile farlo tramite la
              procedura "azienda.sostituisci_responsabile_progetto
              (azienda.get_list_CUP_responsabile_progetto (<vecchiaMatricola>),
              <nuovaMatricola> )", che sostituirà tutte le vecchie occorrenze del
              responsabile con la nuova matricola) e poi potrai procedere con il
              licenziamento.\n I progetti in questione sono:  %', NEW.Matricola,
              lista_responsabile;
        END IF;
    END IF;
    RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER c_tr_licenzia_dipInd_con_incarichi

BEFORE UPDATE ON azienda.DIP_INDETERMINATO

FOR EACH ROW

WHEN (

    OLD.dataFine IS DISTINCT FROM NEW.dataFine OR

    OLD.Tipo <> NEW.Tipo OR

    OLD.Dirigente <> NEW.Dirigente

)

EXECUTE FUNCTION azienda.fn_licenzia_dipInd_con_incarichi();
```

5.8.2 TRIGGERS PER azienda.SCATTO_CARRIERA

nice_looking_domain_sc

Il trigger "nice_looking_domain_sc" rende il campo "Tipo" di "azienda.SCATTO_CARRIERA" case insensitive. Il campo "Tipo" avrà sempre la prima lettera maiuscola e le restanti minuscole.

```
CREATE OR REPLACE FUNCTION azienda.fn_nice_looking_domain_sc()

RETURNS trigger

AS

$$

BEGIN

     NEW.tipo = CONCAT(UPPER(SUBSTR(NEW.tipo, 1, 1)), LOWER(SUBSTR(NEW.Tipo, 2,
        LENGTH(NEW.Tipo))));

     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER a_tr_nice_looking_domain_sc

BEFORE INSERT OR UPDATE OF tipo ON azienda.SCATTO_CARRIERA

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_nice_looking_domain_sc();
```

verifica_scatto_tipo_IU (INSERT & UPDATE)

Il trigger "verifica_scatto_tipo_IU" controlla se lo scatto che si intende inserire (o modificare) rispetti la data esatta in cui dovrebbe avvenire e che il "Tipo" dello scatto in esame sia plausibile rispetto al "Tipo" del dipendente in "DIP_INDETERMINATO". Si noti che viene anche verificato che la matricola inserita effettivamente esista. Questo ha il solo scopo di generare un messaggio di errore personalizzato, dato che tale vincolo è verificato già dal vincolo di chiave esterna.

```
CREATE OR REPLACE FUNCTION azienda.fn_verifica_scatto_tipo_IU()

RETURNS trigger

AS

$$

DECLARE

     dati_matricola RECORD;

BEGIN

     SELECT Tipo, dataAssunzione INTO dati_matricola
     FROM azienda.dip_indeterminato
     WHERE Matricola = NEW.Matricola;


     IF NOT FOUND THEN
         RAISE EXCEPTION 'Matricola % non esistente', NEW.Matricola;
     END IF;


     IF NEW.Tipo = 'Middle' AND dati_matricola.Tipo = 'Junior' THEN
         RAISE EXCEPTION 'La matricola % è "Junior"!  Impossibile registrare lo
            scatto da "Junior" a "Middle"', NEW.Matricola;
     END IF;


     IF NEW.Tipo = 'Middle' AND NEW.Data <> CAST(dati_matricola.dataAssunzione +
        interval '3 years' AS DATE) THEN
         RAISE EXCEPTION 'Per la matricola %, lo scatto da "Junior" a "Middle" deve
            essere in data %!', NEW.Matricola, CAST(dati_matricola.dataAssunzione +
            interval '3 years' AS DATE);
     END IF;


     IF NEW.Tipo = 'Senior' AND (dati_matricola.Tipo = 'Middle' OR
        dati_matricola.Tipo = 'Junior') THEN
         RAISE EXCEPTION 'La matricola % è "Middle"!  Impossibile registrare lo
            scatto da "Middle" a "Senior"', NEW.Matricola;
     END IF;


     IF NEW.Tipo = 'Senior' AND NEW.Data <> CAST(dati_matricola.dataAssunzione +
        interval '7 years' AS DATE) THEN
```

```
        RAISE EXCEPTION 'Per la matricola %, lo scatto da "Middle" a "Senior" deve
          essere in data %!', NEW.Matricola, CAST(dati_matricola.dataAssunzione +
          interval '7 years' AS DATE);
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER b_tr_verifica_scatto_tipo_IU
BEFORE INSERT OR UPDATE ON azienda.SCATTO_CARRIERA
FOR EACH ROW
WHEN (NEW.Tipo = 'Middle' OR NEW.Tipo = 'Senior')
EXECUTE FUNCTION azienda.fn_verifica_scatto_tipo_IU();
```

verifica_scatto_tipo_D (DELETE)

Il trigger "verifica_scatto_tipo_D" ha il compito di verificare se è possibile eliminare uno scatto di carriera in base alla tipologia dell'impiegato e alla tipologia di scatto che si vuole eliminare.

```
CREATE OR REPLACE TRIGGER azienda.fn_verifica_scatto_tipo_D()
RETURNS trigger
AS
$$
DECLARE
    dati_matricola RECORD;
    numero_anni_trascorsi INTEGER := 0;
BEGIN
    SELECT Tipo INTO dati_matricola
    FROM azienda.dip_indeterminato
    WHERE Matricola = OLD.Matricola;

    IF OLD.Tipo = 'Senior' AND dati_matricola.Tipo = 'Senior' THEN
        RAISE EXCEPTION 'La matricola % è "Senior"!  Impossibile eliminare lo
          scatto da "Middle" a "Senior"', OLD.Matricola;
    END IF;

    IF OLD.Tipo = 'Middle' AND (dati_matricola.Tipo = 'Middle' OR
       dati_matricola.Tipo = 'Senior') THEN
```

```
        RAISE EXCEPTION 'La matricola % è "%"!  Impossibile eliminare lo scatto da
          "Junior" a "Middle"', OLD.Matricola, dati_matricola.Tipo;
    END IF;
    RETURN OLD;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER b_tr_verifica_scatto_tipo_D
BEFORE DELETE ON azienda.SCATTO_CARRIERA
FOR EACH ROW
WHEN (OLD.Tipo = 'Middle' OR OLD.Tipo = 'Senior')
EXECUTE FUNCTION azienda.fn_verifica_scatto_tipo_D();
```

verifica_scatto_dirigente_IU (INSERT & UPDATE)

Il trigger "verifica_scatto_dirigente_IU" ha lo scopo di verificare la correttezza dell'operazione di inserimento o aggiornamento di uno scatto dirigenziale all'interno della tabella "azienda.SCATTO_CARRIERA". Controlla la correttezza dell'operazione sulla base di alcune regole, tra cui la coerenza della data dello scatto con la data di inizio e fine del periodo di servizio del dipendente, la corretta sequenza degli scatti dirigenziali e la non molteplicità di più scatti dirigenziali per lo stesso dipendente in una stessa data. Si noti che viene anche verificato che la matricola inserita effettivamente esista. Questo ha il solo scopo di generare un messaggio di errore personalizzato, dato che tale vincolo è verificato già dal vincolo di chiave esterna.

```
CREATE OR REPLACE TRIGGER azienda.fn_verifica_scatto_dirigente_IU()

RETURNS trigger

AS

$$

DECLARE

    cursore_nuova_matricola CURSOR FOR
        SELECT Tipo, Data
        FROM azienda.scatto_carriera
        WHERE Matricola = NEW.Matricola AND (Tipo = 'Rimosso da dirigente' OR Tipo
           = 'Promosso a dirigente')
        ORDER BY Data ASC;


    cursore_vecchia_matricola CURSOR FOR
        SELECT Tipo, Data
        FROM azienda.scatto_carriera
        WHERE Matricola = OLD.Matricola AND (Tipo = 'Rimosso da dirigente' OR Tipo
           = 'Promosso a dirigente')
        ORDER BY Data ASC;
    scatto_attuale RECORD;
    scatto_precedente RECORD;

    dati_matricola RECORD;
    looped BOOLEAN := FALSE;

BEGIN

    SELECT dataAssunzione, dataFine, Dirigente INTO dati_matricola
    FROM azienda.dip_indeterminato
    WHERE Matricola = NEW.Matricola;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'Matricola % non esistente', NEW.Matricola;
    END IF;
```

```
IF OLD.Matricola IS NOT NULL AND NEW.Matricola <> OLD.Matricola THEN
    OPEN cursore_vecchia_matricola;
    FETCH cursore_vecchia_matricola INTO scatto_precedente;
    IF FOUND THEN
        IF scatto_precedente.Tipo <> 'Promosso a dirigente' THEN
           RAISE EXCEPTION 'Cambiando lo scatto di %, il primo scatto risulta
            essere "Rimosso da dirigente"', OLD.Matricola;
        END IF;

        LOOP
           FETCH cursore_vecchia_matricola INTO scatto_attuale;
           EXIT WHEN NOT FOUND;

           IF scatto_precedente.Tipo = scatto_attuale.Tipo THEN
             RAISE EXCEPTION 'Lo scatto in data % è uguale a quello in data %
              per la matricola %', scatto_precedente.Data, scatto_attuale.Tipo,
              OLD.Matricola;
           END IF;

           scatto_precedente = scatto_attuale;
        END LOOP;
    END IF;
    CLOSE cursore_vecchia_matricola;
END IF;
IF dati_matricola.dataFine IS NULL THEN
    IF NEW.Data < dati_matricola.dataAssunzione THEN
        RAISE EXCEPTION 'La matricola % non può essere dirigente (o essere
          rimosso da dirigente) prima di essere assunta!', NEW.Matricola;
    END IF;
ELSE
    IF NEW.Data < dati_matricola.dataAssunzione OR NEW.Data >
      dati_matricola.dataFine THEN
        RAISE EXCEPTION 'La matricola % non può essere dirigente (o essere
          rimosso da dirigente) fuori dal suo periodo di servizio!',
          NEW.Matricola;
    END IF;
END IF;

OPEN cursore_nuova_matricola;
FETCH cursore_nuova_matricola INTO scatto_precedente;

IF scatto_precedente.Tipo <> 'Promosso a dirigente' THEN
    RAISE EXCEPTION 'Il dipendente % non può avere come primo scatto "Rimosso
      da dirigente"!', NEW.Matricola;
```

```
    END IF;
    LOOP
        FETCH cursore_nuova_matricola INTO scatto_attuale;
        EXIT WHEN NOT FOUND;

        IF scatto_precedente.Tipo = scatto_attuale.Tipo THEN
            RAISE EXCEPTION 'Lo scatto in data % è uguale a quello in data % per la
              matricola %', scatto_precedente.Data, scatto_attuale.Data,
              NEW.Matricola;
        ELSIF catto_precedente.Data = scatto_attuale.Data THEN
            RAISE EXCEPTION 'Il dipendente % ha due scatti dirigenziali in data %',
              NEW.Matricola, scatto_precedente.Data;
        END IF;

        scatto_precedente = scatto_attuale;
    END LOOP;
    CLOSE cursore_nuova_matricola;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER a_tr_verifica_scatto_dirigente_IU
BEFORE INSERT OR UPDATE ON azienda.DIP_INDETERMINATO
FOR EACH ROW
WHEN (NEW.Tipo = 'Rimosso da dirigente' OR NEW.Tipo = 'Promosso a dirigente')
EXECUTE FUNCTION azienda.fn_verifica_scatto_dirigente_IU();
```

verifica_scatto_dirigente_D (DELETE)

Il trigger "verifica_scatto_dirigente_D" verifica la coerenza degli scatti di carriera relativi ai dirigenti di un'azienda. In particolare, il trigger viene attivato quando viene effettuata una cancellazione di uno scatto di carriera e la funzione controlla che gli scatti rimanenti per la vecchia matricola del dirigente siano coerenti tra di loro, ovvero che l'alternanza tra "Promosso a dirigente" e "Rimosso da dirigente" sia verificata e che il primo scatto non risulti essere "Rimosso da dirigente". Se viene rilevata una incongruenza, viene generata un'eccezione.

```
CREATE OR REPLACE TRIGGER azienda.fn_verifica_scatto_dirigente_D()
RETURNS trigger
AS
$$
DECLARE
```

```
    cursore_vecchia_matricola CURSOR FOR
        SELECT Tipo, Data
        FROM azienda.scatto_carriera
        WHERE Matricola = OLD.Matricola AND (Tipo = 'Rimosso da dirigente' OR Tipo
          = 'Promosso a dirigente')
        ORDER BY Data ASC;
    scatto_attuale RECORD;
    scatto_precedente RECORD;
BEGIN
    OPEN cursore_vecchia_matricola;
    FETCH cursore_vecchia_matricola INTO scatto_precedente;

    IF FOUND THEN
        IF scatto_precedente.Tipo <> 'Promosso a dirigente' THEN
            RAISE EXCEPTION 'Eliminando lo scatto di %, il primo scatto risulta
              essere "Rimosso da dirigente"', OLD.Matricola;
          END IF;
        LOOP
            FETCH cursore_vecchia_matricola INTO scatto_attuale;
            EXIT WHEN NOT FOUND;

            IF scatto_precedente.Tipo = scatto_attuale.Tipo THEN
               RAISE EXCEPTION 'Lo scatto in data % è uguale a quello in data %
                per la matricola %', scatto_precedente.Data, scatto_attuale.Data,
                OLD.Matricola;
            END IF;

            scatto_precedente = scatto_attuale;
        END LOOP;
    END IF;
    CLOSE cursore_vecchia_matricola;
    RETURN OLD;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER a_tr_verifica_scatto_dirigente_D
AFTER DELETE ON azienda.SCATTO_CARRIERA
FOR EACH ROW
WHEN (OLD.Tipo = 'Rimosso da dirigente' OR OLD.Tipo = 'Promosso a dirigente')
EXECUTE FUNCTION azienda.fn_verifica_scatto_dirigente_D();
```

aggiorna_dirigente_IU (INSERT & UPDATE)

Il trigger "aggiorna_dirigente_IU" gestisce l'aggiornamento dello stato dirigenziale dei dipendenti dell'azienda. In particolare, la funzione verifica lo stato dirigenziale dei dipendenti coinvolti nella transazione e aggiorna la tabella "azienda.DIP_INDETERMINATO" con lo stato dirigenziale aggiornato.

```
CREATE OR REPLACE TRIGGER azienda.fn_aggiorna_dirigente_IU()

RETURNS trigger

AS

$$

DECLARE

    tipo_matricola azienda.scatto_carriera.tipo%TYPE;
    dirigente_matricola azienda.dip_indeterminato.Dirigente%TYPE;

BEGIN

    IF OLD.Matricola IS NOT NULL AND NEW.Matricola <> OLD.Matricola THEN
        SELECT Dirigente INTO dirigente_matricola
        FROM azienda.dip_indeterminato
        WHERE Matricola = OLD.Matricola;

        SELECT Tipo INTO tipo_matricola
        FROM azienda.scatto_carriera
        WHERE Matricola = OLD.Matricola AND
        (Tipo = 'Promosso a dirigente' OR Tipo = 'Rimosso da dirigente')
        ORDER BY Data DESC
        LIMIT 1;

        IF NOT FOUND THEN
            UPDATE azienda.dip_indetermianto
            SET Dirigente = FALSE
            WHERE Matricola = OLD.Matricola;
        ELSE
            IF tipo_matricola = 'Promosso a dirigente' THEN
                IF dirigente_matricola = FALSE THEN
                  UPDATE azienda.dip_indeterminato
                  SET Dirigente = TRUE
                  WHERE matricola = OLD.Matricola;
                END IF;
            ELSE
                IF dirigente_matricola = TRUE THEN
                  UPDATE azienda.dip_indeterminato
                  SET Dirigente = FALSE
                  WHERE matricola = OLD.Matricola;
                END IF;
```

```
            END IF;
        END IF;
    END IF;

    SELECT Tipo INTO tipo_matricola
    FROM azienda.scatto_carriera
    WHERE Matricola = NEW.Matricola AND
    (Tipo = 'Promosso a dirigente' OR Tipo = 'Rimosso da dirigente')
    ORDER BY Data DESC
    LIMIT 1;

    SELECT Dirigente INTO dirigente_matricola
    FROM azienda.dip_indeterminato
    WHERE Matricola = NEW.Matricola;

    IF tipo_matricola = 'Promosso a dirigente' THEN
        IF dirigente_matricola = FALSE THEN
            UPDATE azienda.dip_indeterminato
            SET Dirigente = TRUE
            WHERE matricola = NEW.Matricola;
        END IF;
    ELSE
        IF dirigente_matricola = TRUE THEN
            UPDATE azienda.dip_indeterminato
            SET Dirigente = FALSE
            WHERE matricola = NEW.Matricola;
        END IF;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER b_tr_aggiorna_dirigente_IU
AFTER INSERT OR UPDATE ON azienda.SCATTO_CARRIERA
FOR EACH ROW
WHEN (NEW.Tipo IN ('Promosso a dirigente', 'Rimosso da dirigente'))
EXECUTE FUNCTION azienda.fn_aggiorna_dirigente_IU();
```

aggiorna_dirigente_D (DELETE)

Il trigger "aggiorna_dirigente_D" controlla se un'eliminazione ha riguardato lo stato di dirigente di un dipendente e, in tal caso, aggiorna lo stato di dirigente nella tabella "azienda.DIP_INDETERMINATO"

```
CREATE OR REPLACE TRIGGER azienda.fn_aggiorna_dirigente_D()

RETURNS trigger

AS

$$

DECLARE

    tipo_matricola azienda.scatto_carriera.tipo%TYPE;
    dirigente_matricola azienda.dip_indeterminato.Dirigente%TYPE;

BEGIN

    SELECT Dirigente INTO dirigente_matricola
    FROM azienda.dip_indeterminato
    WHERE Matricola = OLD.Matricola;

    SELECT Tipo INTO tipo_matricola
    FROM azienda.scatto_carriera
    WHERE Matricola = OLD.Matricola AND
    (Tipo = 'Promosso a dirigente' OR Tipo = 'Rimosso da dirigente')
    ORDER BY Data DESC
    LIMIT 1;

    IF NOT FOUND THEN
        UPDATE azienda.dip_indeterminato
        SET Dirigente = FALSE
        WHERE Matricola = OLD.Matricola;
    ELSE
        IF tipo_matricola = 'Promosso a dirigente' THEN
            IF dirigente_matricola = FALSE THEN
                UPDATE azienda.dip_indeterminato
                SET Dirigente = TRUE
                WHERE matricola = OLD.Matricola;
            END IF;
        ELSE
            IF dirigente_matricola = TRUE THEN
                UPDATE azienda.dip_indeterminato
                SET Dirigente = FALSE
                WHERE matricola = OLD.Matricola;
            END IF;
        END IF;
```

```
    END IF;
    RETURN OLD;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER b_tr_aggiorna_dirigente_D
AFTER DELETE ON azienda.SCATTO_CARRIERA
FOR EACH ROW
WHEN (OLD.Tipo IN ('Promosso a dirigente', 'Rimosso da dirigente'))
EXECUTE FUNCTION azienda.fn_aggiorna_dirigente_D();
```

5.8.3 TRIGGERS PER azienda.LABORATORIO

res_scientifico_senior

Il trigger "res_scientifico_senior" ha il compito di verificare se il dipendente che viene assegnato come Responsabile Scientifico per un laboratorio è di tipo "Senior" e se è attualmente in servizio. In caso contrario, viene generata un'eccezione. Si noti che viene anche verificato che la matricola inserita effettivamente esista. Questo ha il solo scopo di generare un messaggio di errore personalizzato, dato che tale vincolo è verificato già dal vincolo di chiave esterna.

```
CREATE OR REPLACE TRIGGER azienda.fn_res_scientifico_senior()

RETURNS trigger

AS

$$

DECLARE

     dip_ind RECORD;

BEGIN

     SELECT Nome, Cognome, Matricola, Tipo, DataFine INTO dip_ind

     FROM azienda.DIP_INDETERMINATO

     WHERE Matricola = NEW.Responsabile_Scientifico;

     IF dip_ind.Matricola IS NULL THEN

         RAISE EXCEPTION 'La matricola % non esiste', NEW.Responsabile_Scientifico;

     END IF;


     IF dip_ind.Tipo <> 'Senior' THEN

         RAISE EXCEPTION 'Il dipendente "% %" con matricola % non è di tipo
            "Senior"!  Non può essere nominato Responsabile scientifico per il
            laboratorio %', dip_ind.Nome, dip_ind.Cognome, dip_ind.Matricola,
            NEW.Nome;

     END IF;


     IF dip_ind.DataFine IS NOT NULL THEN

         RAISE EXCEPTION 'Il dipendente "% %" con matricola % ha una data di
            licenziamento!  Non è possibile assegnare l''incarico di Responsabile
            Scientifico del laboratorio %!', dip_ind.Nome, dip_ind.Cognome,
            dip_ind.Matricola, NEW.Nome;

     END IF;


     RETURN NEW;

END;

$$

LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER tr_res_scientifico_senior
BEFORE INSERT OR UPDATE OF Responsabile_Scientifico
ON azienda.LABORATORIO
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_res_scientifico_senior();
```

aggiungi_lab_res_scient_I (INSERT)

Il trigger "aggiungi_lab_res_scient_I" aggiunge automaticamente l'afferenza del responsabile scientifico di un nuovo laboratorio.

```
CREATE OR REPLACE TRIGGER azienda.fn_aggiungi_lab_res_scient_I()

RETURNS trigger

AS

$$

BEGIN

     INSERT INTO azienda.AFFERIRE(Matricola, nomeLab)
     VALUES (NEW.Responsabile_scientifico, NEW.nome);


     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_z_aggiungi_lab_res_scient_I

AFTER INSERT ON azienda.LABORATORIO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_aggiungi_lab_res_scient_I();
```

aggiungi_lab_res_scient_non_afferente_U (UPDATE)

Il trigger "aggiungi_lab_res_scient_non_afferente_U" verifica se il nuovo Responsabile Scientifico del laboratorio non è già afferente al laboratorio stesso. Se non lo è, la funzione sostituisce l'attuale Responsabile Scientifico del laboratorio con il nuovo Responsabile Scientifico nella tabella "azienda.AFFERIRE".

```
CREATE OR REPLACE TRIGGER azienda.fn_aggiungi_lab_res_scient_non_afferente_U()

RETURNS trigger

AS

$$

BEGIN

     IF NOT EXISTS (SELECT *
     FROM azienda.AFFERIRE
     WHERE Matricola = NEW.Responsabile_scientifico AND
     nomeLab = OLD.nome) THEN
          UPDATE azienda.AFFERIRE
          SET Matricola = NEW.Responsabile_Scientifico
          WHERE Matricola = OLD.Responsabile_Scientifico AND nomeLab = OLD.nome;

          RAISE NOTICE 'L''afferenza del vecchio responsabile scientifico % è stata
             sostituita con l''afferenza del nuovo responsabile scientifico % per il
             laboratorio %
          Non è stata conservata l''afferenza del vecchio responsabile scientifico
             %, che ora non afferirà più al laboratorio %',
             OLD.Responsabile_Scientifico, NEW.Responsabile_Scientifico, NEW.nome,
             OLD.Responsabile_Scientifico, NEW.nome;
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_z_aggiungi_lab_res_scient_non_afferente_U

BEFORE UPDATE OF Responsabile_Scientifico

ON azienda.LABORATORIO

FOR EACH ROW

WHEN (OLD.nome <> NEW.nome OR OLD.Responsabile_scientifico <>
    NEW.Responsabile_scientifico)

EXECUTE FUNCTION azienda.fn_aggiungi_lab_res_scient_non_afferente_U();
```

aggiungi_lab_res_scient_afferente_U (UPDATE)

Il trigger "aggiungi_lab_res_scient_afferente_U" ha l'obiettivo di mantenere aggiornate le afferenze al laboratorio in seguito alla modifica del Responsabile Scientifico con un dipendente già afferente al laboratorio. In tal caso, infatti, è necessario solo eliminare l'afferenza del vecchio Responsabile Scientifico.

```
CREATE OR REPLACE TRIGGER azienda.fn_aggiungi_lab_res_scient_afferente_U()

RETURNS trigger

AS

$$

BEGIN

     IF NOT EXISTS (SELECT *
     FROM azienda.AFFERIRE
     WHERE Matricola = OLD.Responsabile_scientifico AND
     nomeLab = OLD.nome) THEN
          DELETE FROM azienda.AFFERIRE
          WHERE Matricola = OLD.Responsabile_Scientifico AND nomeLab = OLD.nome;

          RAISE NOTICE 'L''afferenza del vecchio responsabile scientifico % è stata
             sostituita con l''afferenza del nuovo responsabile scientifico % per il
             laboratorio %
          Non è stata conservata l''afferenza del vecchio responsabile scientifico
             %, che ora non afferirà più al laboratorio %',
             OLD.Responsabile_Scientifico, NEW.Responsabile_Scientifico, NEW.nome,
             OLD.Responsabile_Scientifico, NEW.nome;
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_z_aggiungi_lab_res_scient_afferente_U

AFTER UPDATE OF Responsabile_Scientifico

ON azienda.LABORATORIO

FOR EACH ROW

WHEN (OLD.nome <> NEW.nome OR OLD.Responsabile_scientifico <>
    NEW.Responsabile_scientifico)

EXECUTE FUNCTION azienda.fn_aggiungi_lab_res_scient_afferente_U();
```

blocco_nAfferenti

Il trigger "blocco_nAfferenti" blocca le modifiche dirette al campo "nAfferenti" della tabella "azienda.LABORATORIO".

```
CREATE OR REPLACE FUNCTION azienda.fn_blocco_nAfferenti()

RETURNS trigger

AS

$$

BEGIN

     IF OLD.nome IS NULL THEN
          IF NEW.nAfferenti <> 1 THEN
              RAISE NOTICE 'Non è possibile inserire direttamente il numero di
                afferenti al laboratorio
              NEW.nAfferenti = 1;
          END IF;


     ELSE
          IF NEW.nAfferenti <> OLD.nAfferenti THEN
              RAISE NOTICE 'Non è possibile modificare direttamente il numero di
                afferenti al laboratorio
              NEW.nAfferenti = OLD.nAfferenti;
          END IF;
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_blocco_nAfferenti

BEFORE INSERT OR UPDATE OF nAfferenti

ON azienda.LABORATORIO

FOR EACH ROW

WHEN (pg_trigger_depth() < 1)

EXECUTE FUNCTION azienda.fn_blocco_nAfferenti();
```

5.8.4 TRIGGERS PER azienda.AFFERIRE

verifica_afferenza

Il trigger "verifica_afferenza" verifica che la matricola inserita esista e sia assegnata ad un dipendente a tempo indeterminato senza data di licenziamento e che il laboratorio esista.
Da notare che i controlli riguardanti l'esistenza della matricola e del laboratorio sono effettuati solo ed esclusivamente per personalizzare il messaggio di errore.

```
CREATE OR REPLACE FUNCTION azienda.fn_verifica_afferenza()

RETURNS TRIGGER

AS

$$

BEGIN

     IF NOT EXISTS (SELECT *
                    FROM azienda.DIP_INDETERMINATO
                    WHERE Matricola = NEW.Matricola) THEN

         RAISE EXCEPTION 'La matricola % non esiste!', NEW.Matricola;
     END IF;


     IF NOT EXISTS (SELECT *
                    FROM azienda.LABORATORIO
                    WHERE nome = NEW.nomeLab) THEN

         RAISE EXCEPTION 'Il laboratorio % non esiste!', NEW.nomeLab;
     END IF;


     IF (SELECT dataFine
         FROM azienda.DIP_INDETERMINATO
         WHERE Matricola = NEW.Matricola) IS NOT NULL THEN

         RAISE EXCEPTION 'La matricola % ha una data di licenziamento!  Non è
            possibile assegnare la nuova afferenza al laboratorio %.',
            NEW.Matricola, NEW.nomeLab;
     END IF;


     RETURN NEW;

END;

$$

LANGUAGE PLPGSQL;
```

```
CREATE OR REPLACE TRIGGER tr_verifica_afferenza

BEFORE INSERT ON azienda.AFFERIRE

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_verifica_afferenza();
```

blocco_aff_lab_res_scient

Il trigger "blocco_aff_lab_res_scient" garantisce che non venga eliminata o aggiornata accidentalmente l'afferenza di un responsabile scientifico di un laboratorio quando si sta operando sulla relazione "azienda.AFFERIRE".
In particolare, il trigger verificherà che non si stia tentando di rimuovere o modificare l'afferenza di un responsabile scientifico di un laboratorio senza prima aver eliminato il laboratorio corrispondente o aver sostituito la matricola del responsabile scientifico per il laboratorio in questione.
La condizione "*WHEN pg_trigger_depth() < 1*" serve ad assicurare che il trigger venga attivato solo quando si sta eliminando o aggiornando una tupla direttamente dalla relazione azienda.AFFERIRE. Questo previene l'attivazione del trigger durante l'esecuzione di altri trigger e garantisce che il controllo sulla matricola venga effettuato correttamente.

```
CREATE OR REPLACE FUNCTION azienda.fn_blocco_aff_lab_res_scient()

RETURNS TRIGGER

AS

$$

BEGIN

     IF EXISTS (SELECT *
                FROM azienda.LABORATORIO
                WHERE Responsabile_scientifico = OLD.Matricola AND
                      nome = OLD.nomeLab) THEN

         RAISE EXCEPTION 'Non è possibile eliminare l''afferenza del responsabile
                          scientifico % del laboratorio %!',
                          OLD.Matricola, OLD.nomeLab;
     END IF;


     RETURN OLD;

END;

$$

LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER tr_blocco_aff_lab_res_scient

BEFORE DELETE OR UPDATE OF Matricola, nomeLab

ON azienda.AFFERIRE

FOR EACH ROW

WHEN (pg_trigger_depth() < 1)

EXECUTE FUNCTION azienda.fn_blocco_aff_lab_res_scient();
```

change_nAfferenti_I (INSERT)

Il trigger "change_nAfferenti_I" verifica che, dopo la registrazione dell'afferenza di un dipendente a tempo indeterminato ad un laboratorio, la matricola del dipendente appena inserita non sia quella del responsabile scientifico di quel laboratorio, al fine di aggiornare correttamente il numero di afferenti ad un laboratorio.
Se così fosse, non deve essere incrementato il numero di afferenti del laboratorio, poichè l'afferenza del responsabile scientifico è già conteggiata; altrimenti incrementa il numero di afferenti.

```
CREATE OR REPLACE FUNCTION azienda.fn_change_nAfferenti_I()

RETURNS TRIGGER

AS

$$

BEGIN

     IF NEW.Matricola = (SELECT Responsabile_scientifico
                         FROM azienda.LABORATORIO
                         WHERE nome = NEW.nomeLab) THEN

         RAISE NOTICE 'L''afferenza del Responsabile Scientifico % di %
                      è già conteggiata!', NEW.Matricola, NEW.nomeLab;
     ELSE

         UPDATE azienda.LABORATORIO
         SET nAfferenti = nAfferenti + 1
         WHERE nome = NEW.nomeLab;

         RAISE NOTICE 'Incrementato di 1 il numero di afferenti
                      del laboratorio %!', NEW.nomeLab;
     END IF;


     RETURN NEW;

END;

$$

LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER tr_change_nAfferenti_I

AFTER INSERT

ON azienda.AFFERIRE

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_change_nAfferenti_I();
```

change_nAfferenti_U (UPDATE)

Il trigger "change_nAfferenti_U" verifica che ci sia coerenza, in una tupla di "azienda.LABORATORIO", tra l'attributo "nAfferenti" e il numero di afferenze al laboratorio stesso, dopo eventuali aggiornamenti. In particolare, data una tupla "(Matricola, nomeLab)" in "azienda.AFFERIRE", vengono analizzate le seguenti circostanze:

1. Nel caso venga aggiornata solo la matricola, il laboratorio non perde alcuna afferenza siccome è rimasto lo stesso, pertanto nAfferenti rimane invariato
2. Nel caso venga aggiornato solo il laboratorio, il vecchio laboratorio perde una afferenza, mentre il nuovo la guadagna
3. Nel caso vengano aggiornati entrambi, il vecchio laboratorio perderà una afferenza, mentre il nuovo laboratorio ne guadagna una.

```
CREATE OR REPLACE FUNCTION azienda.fn_change_nAfferenti_U()

RETURNS TRIGGER

AS

$$

BEGIN

     IF EXISTS (SELECT *
                FROM azienda.LABORATORIO
                WHERE nome = OLD.nomeLab) THEN

          UPDATE azienda.LABORATORIO
          SET nAfferenti = nAfferenti - 1
          WHERE nome = OLD.nomeLab;

          RAISE NOTICE 'Decrementato di 1 il numero di afferenti del vecchio
                        laboratorio %!', OLD.nomeLab;

          UPDATE azienda.LABORATORIO
          SET nAfferenti = nAfferenti + 1
          WHERE nome = NEW.nomeLab;

          RAISE NOTICE 'Incrementato di 1 il numero di afferenti del nuovo
                        laboratorio %!', NEW.nomeLab;
     END IF;


     RETURN NEW;

END;

$$

LANGUAGE PLPGSQL;
```

```
CREATE OR REPLACE TRIGGER tr_change_nAfferenti_U

AFTER UPDATE

ON azienda.AFFERIRE

FOR EACH ROW

WHEN (NEW.nomeLab <> OLD.nomeLab)

EXECUTE FUNCTION azienda.fn_change_nAfferenti_U();
```

change_nAfferenti_D (DELETE)

Il trigger "change_nAfferenti_D" verifica che, per ogni rimozione di una afferenza per un determinato laboratorio, si decrementi il numero di afferenti del laboratorio in questione.

```
CREATE OR REPLACE FUNCTION azienda.fn_change_nAfferenti_D()

RETURNS TRIGGER

AS

$$

BEGIN

     UPDATE azienda.LABORATORIO

     SET nAfferenti = nAfferenti - 1

     WHERE nome = OLD.nomeLab;


     RAISE NOTICE 'Decrementato di 1 il numero di afferenti del laboratorio %!',
        OLD.nomeLab;


     RETURN OLD;

END;

$$

LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER tr_change_nAfferenti_D

AFTER DELETE

ON azienda.AFFERIRE

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_change_nAfferenti_D();
```

5.8.5 TRIGGERS PER azienda.PROGETTO

ref_scientifico_senior

Il trigger "ref_scientifico_senior" verifica che, quando si inserisce un progetto oppure si aggiorna l'attributo "Referente_scientifico" per un progetto già esistente, il nuovo referente scientifico sia di tipo "Senior". Si noti il controllo sulla matricola del nuovo referente scientifico, fatto solo per personalizzare il messaggio di errore.

```
CREATE OR REPLACE FUNCTION azienda.fn_ref_scientifico_senior()

RETURNS TRIGGER

AS

$$

DECLARE

    dip_ind RECORD;

BEGIN

    SELECT Nome, Cognome, Matricola, Tipo, DataFine INTO dip_ind
    FROM azienda.DIP_INDETERMINATO
    WHERE Matricola = NEW.Referente_Scientifico;


    IF dip_ind.Matricola IS NULL THEN

        RAISE EXCEPTION 'La matricola % non esiste', NEW.Referente_Scientifico;
    END IF;


    IF dip_ind.Tipo <> 'Senior' THEN

        RAISE EXCEPTION 'Il dipendente "% %" con matricola % non è di tipo
                        "Senior"!  Non è stato possibile assegnarlo come
                        referente scientifico del progetto %',
                        dip_ind.Nome, dip_ind.Cognome, dip_ind.Matricola, NEW.CUP;
    END IF;


    IF NEW.dataFine IS NULL THEN

        IF dip_ind.DataFine IS NOT NULL THEN

            RAISE EXCEPTION 'Il dipendente "% %" con matricola % ha una data
                            di licenziamento!  Non è possibile assegnare
                            l''incarico di Referente Scientifico del progetto %'
                            dip_ind.Nome, dip_ind.Cognome, dip_ind.Matricola,
                            NEW.CUP;
        END IF;
    ELSE
```

```
        IF dip_ind.DataFine IS NOT NULL AND dip_ind.DataFine < NEW.dataFine THEN
            RAISE EXCEPTION 'Il dipendente "% %" con matricola % è stato licenziato
                             prima della fine del progetto!  Non è stato possibile
                             assegnarlo come referente scientifico del progetto %',
                             dip_ind.Nome, dip_ind.Cognome, dip_ind.Matricola,
                             NEW.CUP;
        END IF;
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;
CREATE OR REPLACE TRIGGER tr_ref_scientifico_senior
BEFORE INSERT OR UPDATE OF Referente_Scientifico
ON azienda.PROGETTO
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_ref_scientifico_senior();
```

res_progetto_dirigente

Il trigger "res_progetto_dirigente" verifica che, quando si inserisce un progetto oppure si aggiorna l'attributo "Responsabile" per un progetto già esistente, il nuovo responsabile sia un dirigente. Si noti il controllo sulla matricola del nuovo responsabile, fatto solo per personalizzare il messaggio di errore.

```
CREATE OR REPLACE FUNCTION azienda.fn_res_progetto_dirigente()

RETURNS TRIGGER

AS

$$

DECLARE

    dip_ind RECORD;

BEGIN

    SELECT Nome, Cognome, Matricola, Dirigente, DataFine INTO dip_ind
    FROM azienda.DIP_INDETERMINATO
    WHERE Matricola = NEW.Responsabile;


    IF dip_ind.Matricola IS NULL THEN

        RAISE EXCEPTION 'La matricola % non esiste', NEW.Responsabile;
    END IF;


    IF dip_ind.Dirigente <> TRUE THEN

        RAISE EXCEPTION 'Il dipendente "% %" con matricola % non è "Dirigente"!
                        Non è stato possibile assegnarlo come responsabile
                         del progetto %', dip_ind.Nome, dip_ind.Cognome,
                        dip_ind.Matricola, NEW.CUP;
    END IF;


    IF NEW.dataFine IS NULL THEN

        IF dip_ind.DataFine IS NOT NULL THEN

            RAISE EXCEPTION 'Il dipendente "% %" con matricola % ha una data
                            di licenziamento!  Non è possibile assegnare
                            l''incarico di Responsabile al progetto %'
                            dip_ind.Nome, dip_ind.Cognome, dip_ind.Matricola,
                            NEW.CUP;
        END IF;
    ELSE
```

```
        IF dip_ind.DataFine IS NOT NULL AND dip_ind.DataFine < NEW.dataFine THEN
            RAISE EXCEPTION 'Il dipendente "% %" con matricola % è stato licenziato
                            prima della fine del progetto!  Non è stato possibile
                            assegnarlo come responsabile del progetto %',
                            dip_ind.Nome, dip_ind.Cognome, dip_ind.Matricola,
                            NEW.CUP;
        END IF;
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;
CREATE OR REPLACE TRIGGER tr_res_progetto_dirigente
BEFORE INSERT OR UPDATE OF Responsabile
ON azienda.PROGETTO
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_res_progetto_dirigente();
```

blocco_modifiche_budget_costi

Il trigger "blocco_modifiche_budget_costi" verifica che venga impedita qualsiasi modifica diretta agli attributi "costoAttrezzature" e "costoContrattiProgetto" per una tupla della relazione "azienda.PROGETTO" La condizione "*WHEN pg_trigger_depth() <1*" serve per innescare il trigger ogni qual volta si intende inserire o aggiornare una tupla direttamente dalla relazione azienda.PROGETTO, evitando così che il bloccco sull'inserimento o l'aggiornamento si presenti durante l'azione di trigger innescati prima di esso.

```
CREATE OR REPLACE FUNCTION azienda.fn_blocco_modifiche_budget_costi()

RETURNS TRIGGER

AS

$$

DECLARE

    half_budget azienda.EURO := (0.5 * NEW.Budget)-0.005;

BEGIN

    IF OLD.CostoAttrezzature IS NULL THEN

        IF (NEW.costoAttrezzature <> 0 OR NEW.costoContrattiProgetto <> 0) THEN

            RAISE NOTICE E'Non puoi inserire manualmente il costo totale delle
                          attrezzature o il costo totale dei contratti
                          a progetto di %!\nBisogna comprare le attrezzature
                          o i contratti relativi a questo progetto.\n
                          L''inserimento a questi campi è stato ignorato',
                          NEW.CUP;
        END IF;
    ELSE
        IF (NEW.CostoAttrezzature <> OLD.costoAttrezzature OR
            NEW.costoContrattiProgetto <> OLD.costoContrattiProgetto) THEN

            RAISE NOTICE E'Non puoi modificare manualmente il costo totale delle
                          attrezzature o il costo totale dei contratti
                          a progetto di %!\nBisogna modificare gli acquisti
                          delle attrezzature o i contratti
                          relativi a questo progetto.\n
                          Le modifiche di questi campi sono state ignorate',
                          NEW.CUP;
        END IF;
    END IF;


    RETURN NEW;

END;

$$
```

```
LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER tr_blocco_modifiche_budget_costi

BEFORE INSERT OR UPDATE OF costoAttrezzature, costoContrattiProgetto ON
    azienda.PROGETTO

FOR EACH ROW

WHEN (pg_trigger_depth() < 1)

EXECUTE FUNCTION azienda.fn_blocco_modifiche_budget_costi();
```

verifica_budget_costi

Il trigger "verifica_budget_costi" blocca la modifica del "Budget" nel caso l'attuale "costoAttrezzature" o "costoContrattiProgetto" sforino il limite del 50% del nuovo "Budget".

```
CREATE OR REPLACE FUNCTION azienda.fn_verifica_budget_costi()

RETURNS TRIGGER

AS

$$

DECLARE

     half_budget azienda.EURO := (0.5 * NEW.Budget)-0.005;

BEGIN

     IF ((half_budget < NEW.CostoAttrezzature) OR
          (half_budget < NEW.costoContrattiProgetto)) THEN

          RAISE EXCEPTION 'Non è stato possibile modificare il Budget da % a %
                          perchè attualmente per il progetto % vi è una spesa
                          in attrezzature o contratti a progetto superiore
                          a % (il 50%% di %)!  Modifica rifiutata.',
                          OLD.Budget, NEW.Budget, NEW.CUP, half_budget, NEW.Budget;
     END IF;


     RETURN NEW;

END;

$$

LANGUAGE PLPGSQL;

CREATE OR REPLACE TRIGGER tr_verifica_budget_costi

BEFORE UPDATE OF Budget ON azienda.PROGETTO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_verifica_budget_costi();
```

5.8.6 TRIGGERS PER azienda.LAVORARE

controllo_laboratori_progetto

Il trigger "controllo_laboratori_progetto" controlla la correttezza e la coerenza dei dati inseriti nella tabella "azienda.LAVORARE". In particolare, controlla che il numero massimo di laboratori assegnati ad un progetto (cioè massimo 3 laboratori) non sia superato. Vengono effettuati ulteriori controlli sull'esistenza del progetto o del laboratorio con il solo scopo di personalizzare il messaggio di errore rispetto a quello che già darebbe per violazione di vincolo di chiave esterna o di chiave primaria.

```
CREATE OR REPLACE FUNCTION azienda.fn_controllo_laboratori_progetto()

RETURNS trigger

AS

$$

DECLARE

     lavora_su RECORD;
     cup_exists RECORD;
     laboratorio_exists RECORD;
     numLabOnProg INTEGER := 0;

BEGIN

     SELECT * INTO cup_exists
     FROM azienda.PROGETTO
     WHERE CUP = NEW.CUP;

     IF NOT FOUND THEN
         RAISE EXCEPTION 'Il progetto con CUP % non esiste', NEW.CUP;
     END IF;

     SELECT * INTO laboratorio_exists
     FROM azienda.LABORATORIO
     WHERE nome = NEW.nomeLab;

     IF NOT FOUND THEN
         RAISE EXCEPTION 'Il laboratorio % non esiste', NEW.nomeLab;
     END IF;

     SELECT * INTO lavora_su
     FROM azienda.LAVORARE
     WHERE CUP = NEW.CUP AND nomeLab = NEW.nomeLab;

     IF FOUND THEN
```

```
        RAISE EXCEPTION 'Il laboratorio % lavora già sul progetto con CUP %',
          NEW.nomeLab, NEW.CUP;
    END IF;

    SELECT COUNT(*) INTO numLabOnProg
    FROM azienda.LAVORARE
    WHERE CUP = NEW.CUP AND
    nomeLab NOT IN(
        SELECT nomeLab
        FROM azienda.LAVORARE
        WHERE CUP = OLD.CUP AND nomeLab = OLD.nomeLab
    )
    IF numLabOnProg >= 3 THEN
        RAISE EXCEPTION 'Non è possibile assegnare più di 3 laboratori ad un
          progetto!  Il laboratorio % non può essere aggiunto al progetto %',
          NEW.nomeLab, NEW.CUP;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER tr_controllo_laboratori_progetto
BEFORE INSERT OR UPDATE ON azienda.LAVORARE
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_controllo_laboratori_progetto();
```

5.8.7 TRIGGERS PER azienda.ATTREZZATURA

coerenza_acquisto_attr_lab

Il trigger "coerenza_acquisto_attr_lab" verifica che il laboratorio a cui si vuole assegnare un'attrezzatura lavori effettivamente per il progetto tramite cui l'attrezzatura è stata acquistata. Viene inoltre verificato che il laboratorio e il progetto specificati esistano effettivamente. Ciò ha solo finalità di messaggio di errore personalizzato.

```
CREATE OR REPLACE FUNCTION azienda.fn_coerenza_acquisto_attr_lab()

RETURNS trigger

AS

$$

DECLARE

     corrispondenze RECORD;

BEGIN

     IF NOT EXISTS (SELECT *
     FROM azienda.LABORATORIO
     WHERE nome = NEW.nomeLab) THEN
         RAISE EXCEPTION 'Il laboratorio % non esiste!', NEW.nomeLab;
     END IF;

     IF NOT EXISTS (SELECT *
     FROM azienda.PROGETTO
     WHERE CUP = NEW.CUP) THEN
         RAISE EXCEPTION 'Il progetto % non esiste!', NEW.CUP;
     END IF;

     IF NOT EXISTS (SELECT *
     FROM azienda.LAVORARE
     WHERE CUP = NEW.CUP AND nomeLab = NEW.nomeLab) THEN
         RAISE EXCEPTION 'Il laboratorio % non lavora per il progetto %, % non
           registrato come attrezzatura!', NEW.nomeLab, NEW.CUP, NEW.Descrizione;
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_coerenza_acquisto_attr_lab

BEFORE INSERT OR UPDATE OF nomeLab, CUP
```

```
ON azienda.ATTREZZATURA

FOR EACH ROW

WHEN (NEW.nomeLab IS NOT NULL AND NEW.CUP IS NOT NULL)

EXECUTE FUNCTION azienda.fn_coerenza_acquisto_attr_lab();
```

incr_costo_attrezzature_e_50_I (INSERT)

Il trigger "incr_costo_attrezzature_e_50_I" controlla l'aggiunta di una nuova attrezzatura ad un progetto dell'azienda, verificando che il costo totale delle attrezzature non superi il 50% del budget del progetto, se la condizione è verificata il trigger aggiorna il costo totale delle attrezzature del progetto con il nuovo valore. In caso contrario l'attrezzatura non viene inserita.

```
CREATE OR REPLACE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_I()

RETURNS trigger

AS

$$

DECLARE

     dati_progetto record;
     half_budget azienda.EURO := 0;

BEGIN

     SELECT Budget, costoAttrezzature INTO dati_progetto
     FROM azienda.PROGETTO
     WHERE CUP = NEW.CUP;


     dati_progetto.CostoAttrezzature = dati_progetto.CostoAttrezzature + NEW.Costo;
     half_budget := (0.5  dati_progetto.Budget)-0.005;


     IF half_budget >= dati_progetto.CostoAttrezzature THEN
          UPDATE azienda.PROGETTO AS PR
          SET costoAttrezzature = dati_progetto.CostoAttrezzature
          WHERE PR.CUP = NEW.CUP;
     ELSE
          RAISE EXCEPTION 'Il costo totale delle attrezzature, ovvero €%, sfora il
            50%% del budget (€%)!  L''attrezzatura "%" non è stata acquistata per
            il progetto %', dati_progetto.CostoAttrezzature, half_budget,
            NEW.Descrizione, NEW.CUP;
     END IF;
     RETURN NEW;

END;

$$
```

```
LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_incr_costo_attrezzature_e_50_I

BEFORE INSERT ON azienda.ATTREZZATURA

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_I();
```

incr_costo_attrezzature_e_50_U (UPDATE)

Il trigger "incr_costo_attrezzature_e_50_U" verifica se la modifica del costo di un'attrezzatura in un progetto supera il 50% del budget del progetto stesso. In caso contrario, viene aggiornato il costo delle attrezzature nel progetto.

```
CREATE OR REPLACE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_U()

RETURNS trigger

AS

$$

DECLARE

    dati_progetto record;
    half_budget azienda.EURO := 0;

BEGIN

    SELECT Budget, costoAttrezzature INTO dati_progetto
    FROM azienda.PROGETTO
    WHERE CUP = NEW.CUP;


    dati_progetto.CostoAttrezzature = dati_progetto.CostoAttrezzature + NEW.Costo;
    IF OLD.CUP = NEW.CUP THEN
        dati_progetto.CostoAttrezzature = dati_progetto.CostoAttrezzature -
          OLD.Costo;
    END IF;


    half_budget := (0.5  dati_progetto.Budget)-0.005;


    IF half_budget >= dati_progetto.CostoAttrezzature THEN
        UPDATE azienda.PROGETTO AS PR
        SET costoAttrezzature = dati_progetto.CostoAttrezzature
        WHERE PR.CUP = NEW.CUP;

        IF OLD.CUP <> NEW.CUP THEN
            UPDATE azienda.PROGETTO AS PR
            SET costoAttrezzature = costoAttrezzature - OLD.Costo
            WHERE PR.CUP = OLD.CUP;
```

```
        END IF;
    ELSE
        RAISE EXCEPTION 'Il costo totale delle attrezzature, ovvero €%, sfora il
           50%% del budget (€%)!  L''attrezzatura "%" non è stata acquistata per
           il progetto %', dati_progetto.CostoAttrezzature, half_budget,
           NEW.Descrizione, NEW.CUP;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER tr_incr_costo_attrezzature_e_50_U
BEFORE UPDATE ON azienda.ATTREZZATURA
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_U();
```

incr_costo_attrezzature_e_50_D (DELETE)

Il trigger "incr_costo_attrezzature_e_50_D" gestisce la rimozione di attrezzature da un progetto. La funzione decrementa il costo dell'attrezzatura eliminata dal costo totale delle attrezzature del progetto.

```
CREATE OR REPLACE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_D()
RETURNS trigger
AS
$$
DECLARE
    dati_progetto record;
BEGIN
    SELECT Budget, costoAttrezzature INTO dati_progetto
    FROM azienda.PROGETTO
    WHERE CUP = OLD.CUP;

    dati_progetto.CostoAttrezzature = dati_progetto.CostoAttrezzature - OLD.Costo;

    UPDATE azienda.PROGETTO AS PR
    SET costoAttrezzature = dati_progetto.CostoAttrezzature
    WHERE PR.CUP = OLD.CUP;
    RETURN OLD;
END;
```

```
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER tr_incr_costo_attrezzature_e_50_D
AFTER DELETE ON azienda.ATTREZZATURA
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_D();
```

5.8.8 TRIGGERS PER azienda.DIP_PROGETTO

verifica_dipProg

Il trigger "verifica_dipProg" verificare se la data di scadenza di un dipendente a progetto è compresa tra le date di inizio e fine del progetto a cui è associato. Se la data di scadenza non rientra in questo intervallo, viene generata un'eccezione.

```
CREATE OR REPLACE FUNCTION azienda.fn_verifica_dipProg()

RETURNS trigger

AS

$$

DECLARE

     dati_progetto record;

BEGIN

     SELECT dataInizio, dataFine INTO dati_progetto

     FROM azienda.PROGETTO

     WHERE CUP = OLD.CUP;


     IF NEW.Scadenza < dati_progetto.dataInizio OR NEW.Scadenza >
        dati_progetto.dataFine THEN

         RAISE EXCEPTION 'La data di scadenza del dipendente a progetto % non è
            compresa tra %(inizio progetto) e %(fine progetto)!', NEW.Matricola,
            dati_progetto.dataInizio, dati_progetto.dataFine;

     END IF;


     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_verifica_dipProg

BEFORE INSERT OR UPDATE OF Scadenza

ON azienda.DIP_PROGETTO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_verifica_dipProg();
```

incr_costo_dipProgetto_e_50_I (INSERT)

Il trigger "incr_costo_dipProgetto_e_50_I" controlla se il costo totale dei contratti di un progetto supera il 50% del budget allocato per tale progetto. In caso contrario, viene aggiornato il costo dei contratti a progetto in "azienda.PROGETTO".

```
CREATE OR REPLACE FUNCTION azienda.fn_incr_costo_dipProgetto_e_50_I()

RETURNS trigger

AS

$$

DECLARE

     dati_progetto record;
     half_budget azienda.EURO := 0;

BEGIN

     SELECT Budget, costoContrattiProgetto INTO dati_progetto
     FROM azienda.PROGETTO
     WHERE CUP = OLD.CUP;


     dati_progetto.costoContrattiProgetto = dati_progetto.costoContrattiProgetto +
        NEW.Costo;
     half_budget := (0.5 * dati_progetto.Budget)-0.005;


     IF half_budget >= dati_progetto.costoContrattiProgetto THEN
         UPDATE azienda.PROGETTO AS PR
         SET costoContrattiProgetto = dati_progetto.costoContrattiProgetto
         WHERE PR.CUP = NEW.CUP;
     ELSE
         RAISE EXCEPTION 'Il costo totale dei contratti a progetto, ovvero €%,
            sfora il 50%% del budget (€%)!  Il contratto % è stato annullato per il
            progetto %', dati_progetto.costoContrattiProgetto, half_budget,
            NEW.Matricola, NEW.CUP;
     END IF;
     RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER tr_incr_costo_dipProgetto_e_50_I

BEFORE INSERT ON azienda.DIP_PROGETTO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_incr_costo_dipProgetto_e_50_I();
```

incr_costo_dipProgetto_e_50_U (UPDATE)

Il trigger "incr_costo_dipProgetto_e_50_U" verifica se la modifica di un contratto associato a un progetto di un'azienda causa il superamento del 50% del budget del progetto stesso. In caso contrario, viene aggiornato il costo dei contratti a progetto in "azienda.PROGETTO".

```
CREATE OR REPLACE FUNCTION azienda.fn_incr_costo_dipProgetto_e_50_U()

RETURNS trigger

AS

$$

DECLARE

    dati_progetto record;
    half_budget azienda.EURO := 0;

BEGIN

    SELECT Budget, costoContrattiProgetto INTO dati_progetto
    FROM azienda.PROGETTO
    WHERE CUP = NEW.CUP;


    dati_progetto.costoContrattiProgetto = dati_progetto.costoContrattiProgetto +
       NEW.Costo;
    IF OLD.CUP = NEW.CUP THEN
        dati_progetto.costoContrattiProgetto = dati_progetto.costoContrattiProgetto
          - OLD.Costo;
    END IF;


    half_budget := (0.5  dati_progetto.Budget)-0.005;


    IF half_budget >= dati_progetto.costoContrattiProgetto THEN
        UPDATE azienda.PROGETTO AS PR
        SET costoContrattiProgetto = dati_progetto.costoContrattiProgetto
        WHERE PR.CUP = NEW.CUP;


        IF OLD.CUP <> NEW.CUP THEN
            UPDATE azienda.PROGETTO AS PR
            SET costoContrattiProgetto = costoContrattiProgetto - OLD.Costo
            WHERE PR.CUP = OLD.CUP;
        END IF;
    ELSE
        RAISE EXCEPTION 'Il costo totale dei contratti a progetto, ovvero €%,
          sfora il 50%% del budget (€%)!  Il contratto % è stato annullato per il
          progetto %', dati_progetto.costoContrattiProgetto, half_budget,
          NEW.Descrizione, NEW.CUP;
```

```
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER tr_incr_costo_dipProgetto_e_50_U
BEFORE UPDATE ON azienda.DIP_PROGETTO
FOR EACH ROW
EXECUTE FUNCTION azienda.fn_incr_costo_dipProgetto_e_50_U();
```

incr_incr_costo_dipProgetto_e_50_D (DELETE)

Il trigger "incr_costo_dipProgetto_e_50_D" gestisce l'eliminazione di record dalla tabella azienda.DIP_PROGETTO. In particolare la funzione associata al trigger aggiorna il valore della colonna "costoContrattiProgetto" nella tabella azienda.PROGETTO, sottraendo il valore del costo dell'elemento eliminato.

```
CREATE OR REPLACE FUNCTION azienda.fn_incr_costo_attrezzature_e_50_D()
RETURNS trigger
AS
$$
DECLARE
    dati_progetto record;
BEGIN
    SELECT Budget, costoContrattiProgetto INTO dati_progetto
    FROM azienda.PROGETTO
    WHERE CUP = OLD.CUP;

    dati_progetto.costoContrattiProgetto = dati_progetto.costoContrattiProgetto -
       OLD.Costo;

    UPDATE azienda.PROGETTO AS PR
    SET costoContrattiProgetto = dati_progetto.costoContrattiProgetto
    WHERE PR.CUP = OLD.CUP;
    RETURN OLD;
END;
$$
LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER tr_incr_costo_dipProgetto_e_50_D
```

```
AFTER DELETE ON azienda.DIP_PROGETTO

FOR EACH ROW

EXECUTE FUNCTION azienda.fn_incr_costo_dipProgetto_e_50_D();
```