UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Corso di Laurea Triennale in Informatica

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

**Mattia Scotellaro N86004846**
**Marta Pagliuso N86004844**

ANNO ACCADEMICO 2022/2023

# Indice


**1 Requisiti identificati** **3**

**2 Progettazione concettuale** **5**
2.1 Diagramma (UML) . . . 5
2.2 Analisi della ristrutturazione del diagramma UML . . . 6
2.2.1 Analisi delle ridondanze . . . 6
2.2.2 Eliminazione delle generalizzazioni . . . 6
2.2.3 Eliminazione degli attributi multivalore . . . 6
2.2.4 Eliminazione degli attributi strutturati . . . 7
2.2.5 Partizione/Accorpamento delle associazioni . . . 7
2.2.6 Scelta degli Identificatori Primari . . . 7
2.3 Diagramma ristrutturato (UML) . . . 8
2.4 Diagramma ristrutturato (ER) . . . 9

**3 Dizionari** **10**
3.1 Dizionario delle Entità . . . 10
3.2 Dizionario delle Associazioni . . . 11
3.3 Dizionario dei Vincoli . . . 12

**4 Modello Logico** **13**
4.1 Traduzione di Entità e Associazioni . . . 13
4.2 Relazioni . . . 13

**5 Modello Fisico** **15**
5.1 Domini . . . 15
5.1.1 Dominio Categoria . . . 15
5.1.2 Dominio Codice Fiscale . . . 15
5.1.3 Dominio Tipo Contratto . . . 15
5.2 Tabelle . . . 16
5.2.1 Impiegato . . . 16
5.2.2 Promozione . . . 16
5.2.3 Laboratorio . . . 17
5.2.4 Progetto . . . 17
5.2.5 Lavora . . . 18
5.2.6 Utilizza . . . 18
5.2.7 Attrezzatura . . . 19
5.3 Viste . . . 19
5.3.1 Lista Attrezzature Progetto . . . 19
5.3.2 Lista Impiegati Progetto . . . 19
5.3.3 Afferenti Progetto . . . 20
5.3.4 Tabulato Dipendenti . . . 20
5.3.5 Tabulato Contratti . . . 20
5.4 Procedure e Funzioni . . . 21
5.4.1 add_progetto . . . 21

5.4.2 add_attrezzatura . . . . . 21
5.4.3 add_laboratorio . . . . . 22
5.4.4 generate_random_salary . . . . . 22
5.4.5 add_impiegato_progetto . . . . . 23
5.4.6 add_impiegato . . . . . 24
5.4.7 delete_impiegato . . . . . 25
5.4.8 substitute_impiegato . . . . . 25
5.4.9 add_promozione . . . . . 26
5.4.10 add_utilizza . . . . . 26
5.4.11 add_lavora . . . . . 27
5.4.12 acquista_attrezzatura . . . . . 27
5.5 Triggers . . . . . 28
5.5.1 check_cat_prog . . . . . 28
5.5.2 check_cat_lab . . . . . 29
5.5.3 check_acquista_attrezzatura . . . . . 29
5.5.4 check_delete_impiegato . . . . . 30
5.5.5 check_add_promozione . . . . . 31
5.5.6 check_add_utilizza . . . . . 32
5.5.7 check_add_impiegato . . . . . 33
5.5.8 check_add_impiegato_progetto . . . . . 34
5.5.9 check_add_lavora . . . . . 35
5.5.10 update_lab . . . . . 36
5.5.11 add_afferente . . . . . 36
5.5.12 remove_afferente . . . . . 37

# 1 Requisiti identificati

In questa sezione, esamineremo la richiesta del cliente con l'obiettivo di delineare le caratteristiche che la base di dati deve possedere. Identificheremo le entità e le relazioni del mini-world al fine di rappresentare in modo completo e comprensibile il dominio del problema.

> *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 più di 3 anni per l'azienda;*
> 3. *Dipendente senior: colui che lavora da almeno 7 anni all'interno dell'azienda;*
> 4. *Dirigente: 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.*

In questa parte viene messa in evidenza la divisione gerarchica dei dipendenti all'interno dell'azienda. Ci sono 3 categorie principali: Junior, Middle e Senior. Un dipendente può salire di livello in base agli anni di servizio conseguiti all'interno dell'azienda. Eccezione fatta per la quarta categoria, ovvero quella del Dirigente, in cui può partecipare un qualsiasi impiegato, a patto che ne abbia le capacità.

> *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.*

Qui vengono specificate due delle entità gestite all'interno dell'azienda. La prima è quella del Laboratorio, caratterizzato da un topic, ovvero in cosa è specializzato; il numero di afferenti, ovvero il numero di impiegati che ci lavorano e il responsabile scientifico che gestisce il laboratorio a lui assegnato con il vincolo che sia 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.*

La seconda entità è quella del Progetto. Questo è identificato dal CUP, un codice alfanumerico di 15 caratteri, e da un nome, entrambi unici nel sistema. Di un Progetto si devono occupare un referente scientifico e un responsabile, rispettivamente un dipendente senior e un dirigente.

*Al massimo 3 laboratori possono lavorare a un progetto.*

Il vincolo in questione è molto chiaro: a ogni Progetto possono essere affidati da un minimo di 1 a un massimo di 3 Laboratori.

*Un laboratorio ha diverse attrezzature (computer, robot, dispositivi mobili, sensori, ...) acquistati con i fondi di un determinato progetto.*

La caratterizzazione dell'entità Laboratorio si fa più precisa con l'aggiunta di una nuova entità: Attrezzatura. Questa non viene descritta in modo specifico come le precedenti. Le sue specifiche possono essere scelte arbitrariamente in base alle esigenze.

*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.*

Qui viene presentato il primo vincolo inter-relazionale che coinvolge le entità Progetto e Attrezzatura. In particolare, l'acquisto di un'Attrezzatura può avvenire un numero limitato di volte, in base al 50% del budget stipulato per 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.*

Gli impiegati dell'azienda sono divisi in due categorie: coloro che possiedono un contratto a tempo indeterminato e coloro che hanno un contratto "a progetto". Ciò significa che il contratto di questi ultimi è valido solo per il periodo necessario alla realizzazione del Progetto a cui è associato. Non vengono fatte altre specifiche, dunque eventuali condizioni sono a discrezione del basista.

# 2 Progettazione concettuale

## 2.1 Diagramma (UML)

Considerando l'analisi dei requisiti svolta nella sezione precedente, viene ora presentata la prima iterazione del Class Diagram.

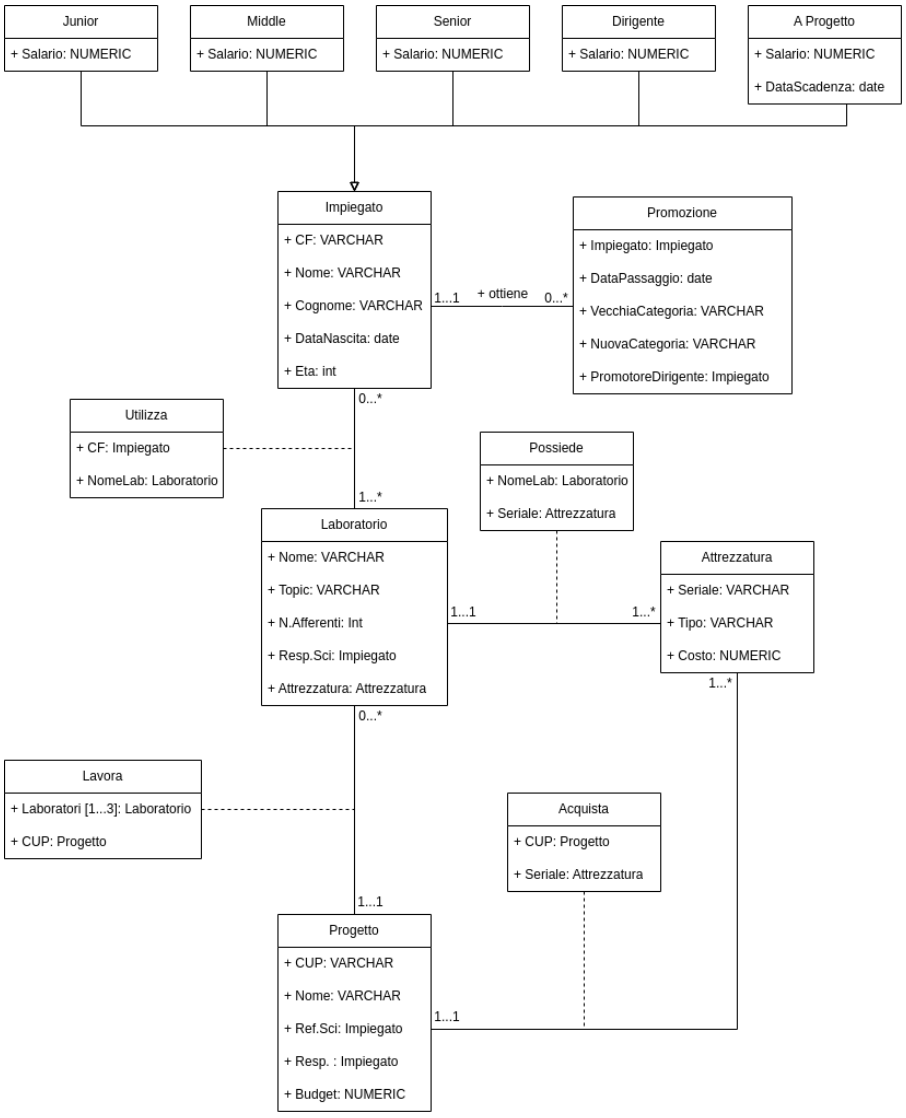


Figura 1: Diagramma UML non ristrutturato

## 2.2 Analisi della ristrutturazione del diagramma UML

### 2.2.1 Analisi delle ridondanze

Una ridondanza, in uno schema concettuale, corrisponde alla presenza di un dato che può essere derivato da altri dati.

Nel nostro caso sono presenti due dati derivabili:

- Impiegato.Eta: serve per indicare l'età dell'impiegato. La si può ricavare tramite una semplice operazione matematica in cui viene sottratto l'anno della data di nascita del suddetto impiegato dall'anno corrente;
- Laboratorio.N.Afferenti: serve per tenere traccia del numero di impiegati che lavorano in un laboratorio. Viene ricavato automaticamente durante la popolazione della tabella Utilizza.

### 2.2.2 Eliminazione delle generalizzazioni

Nel seguente diagramma UML è presente una singola generalizzazione che si dirama in cinque specializzazioni. In particolare, l'entità Impiegato si specializza nelle quattro categorie specificate nella traccia, più quella *a progetto*.

Nei sistemi tradizionali non si possono rappresentare le generalizzazioni. Abbiamo quindi bisogno di tradurle usando altri costrutti della modellazione concettuale.

Nel nostro caso, abbiamo una generalizzazione di tipo disgiunta-totale perché un impiegato non può appartenere a più categorie contemporaneamente, ma almeno ad una di esse. Dunque è stato deciso di risolverla accorpando le classi figlie nel padre e aggiungendo due attributi:

- Categoria: per le categorie ottenute per anzianità di servizio;
- TipoContratto: per distinguere gli impiegati con contratto a tempo indeterminato da quelli "a progetto".

### 2.2.3 Eliminazione degli attributi multivalore

L'unico attributo multivalore presente è *Lavora.Laboratori[1...3]*. Viene usato per rispettare la condizione, identificata durante l'analisi dei requisiti, che impone l'associazione di al più tre laboratori a un singolo progetto.

Siccome il modello relazionale non consente questo tipo di rappresentazione, è stato deciso di risolverlo non considerandolo come multivalore ma come se fosse singolo. Il vincolo resta verificato tramite un conteggio delle istanze di un laboratorio per ogni progetto nella tabella in questione.

### 2.2.4 Eliminazione degli attributi strutturati

Gli attributi strutturati presenti nel Class Diagram in questione sono:

- Promozione.Impiegato;
- Promozione.PromotoreDirigente;
- Utilizza.CF;
- Utilizza.NomeLab;
- Laboratorio.Resp.Sci;
- Laboratorio.Attrezzatura;
- Progetto.Ref.Sci;
- Progetto.Resp;
- Possiede.NomeLab;
- Possiede.Seriale;
- Lavora.Laboratori;
- Lavora.CUP;
- Acquista.CUP;
- Acquista.Seriale;

Sono stati risolti trascurando la struttura dell'attributo e, quindi, sostituendolo con una stringa coerente al contesto.

### 2.2.5 Partizione/Accorpamento delle associazioni

Si procede ora ad eliminare eventuali classi superflue. In questo caso le candidate sono Acquista e Possiede che, presentando solo chiavi esterne e facendo parte di associazioni di tipo [1...1], non sono necessarie ai fini dell'implementazione. Sono stati così aggiunti all'entità Attrezzatura gli attributi *NomeLab* e *CUP*, presi rispettivamente da Possiede e Acquista.

### 2.2.6 Scelta degli Identificatori Primari

In questa sezione si identificano le chiavi primarie di ogni entità principale per il recupero efficiente dei dati.

- **Impiegato**: l'unica chiave candidata è il codice fiscale (CF). È stata scelta perché, tra gli attributi della classe, è l'unico che è univoco per ogni istanza della stessa;

- **Laboratorio**: l'unica chiave candidata è il nome (Nome). Considerando la natura dell'attributo, sarebbe possibile aggiungerne un altro, ma non è questo il caso poiché è stato deciso che i laboratori dell'azienda non possono possedere nomi uguali tra loro.
- **Attrezzatura**: L'unica chiave candidata è il seriale (Seriale). È stata scelta in quanto, nella traccia, non sono presenti indicazioni a riguardo;
- **Progetto**: Per la traccia le chiavi candidate sono il CUP (CUP) e il nome (Nome), unico nel sistema. Tra le due, è stato scelto il CUP.

## 2.3 Diagramma ristrutturato (UML)

Applicate tutte le modifiche riportate nella sezione precedente, questo è il Class Diagram ristrutturato:

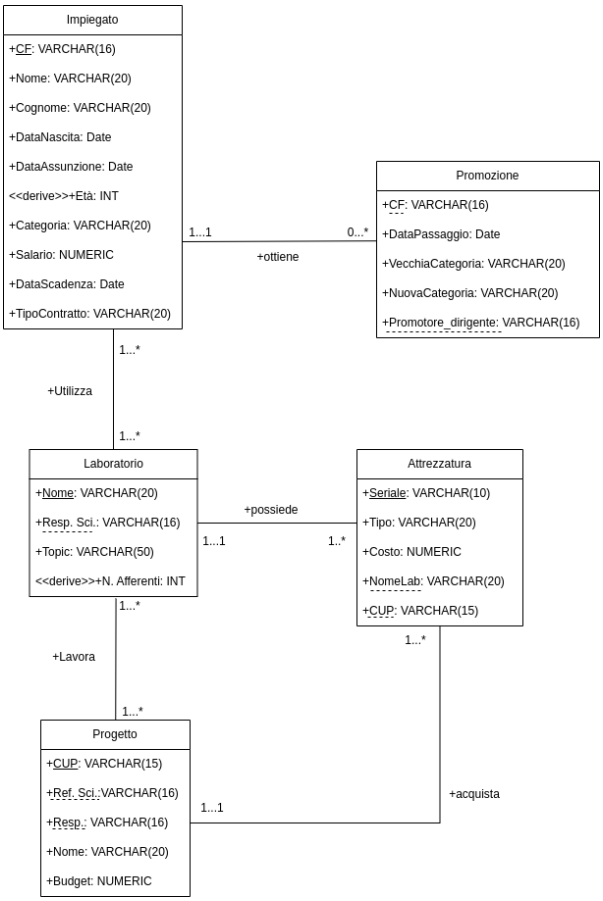


Figura 2: Diagramma UML ristrutturato

## 2.4 Diagramma ristrutturato (ER)

Viene ora presentata un'ulteriore rappresentazione dello schema ristrutturato tramite diagramma ER:

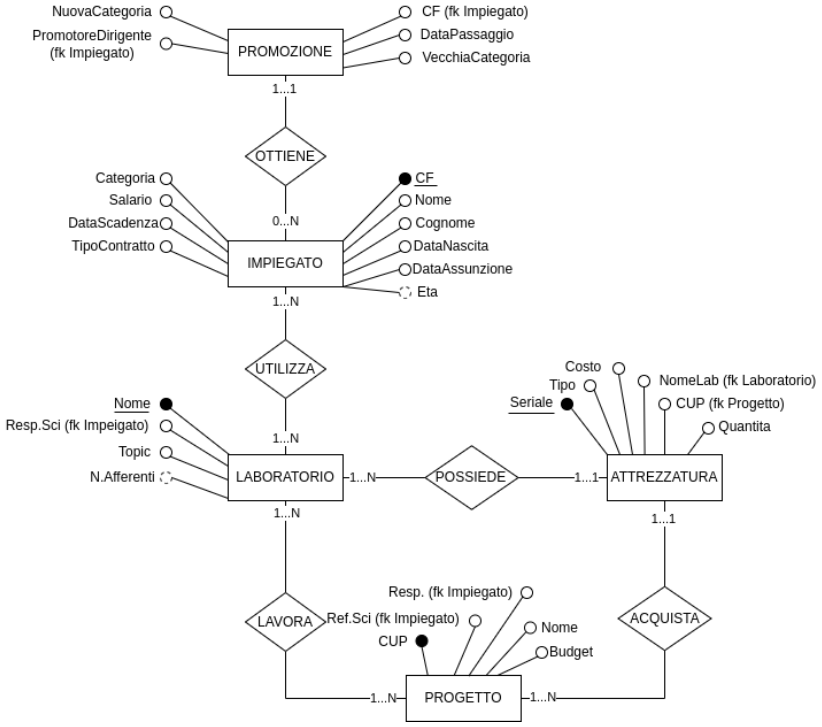


Figura 3: Diagramma ER ristrutturato

# 3 Dizionari

## 3.1 Dizionario delle Entità

| Entità | Descrizione | Attributi |
|---|---|---|
| Impiegato | Entità che tiene traccia degli impiegati dell'azienda | **CF**(Stringa): codice univoco che identifica un impiegato.<br>**Nome**(Stringa): nome di un impiegato.<br>**Cognome**(Stringa): cognome di un impiegato.<br>**DataNascita**(Data): data di nascita di un impiegato.<br>**DataAssunzione**(Data): data di assunzione di un impiegato nell'azienda.<br>**Eta**(Int): età di un impiegato.<br>**Categoria**(Stringa): categoria di un impiegato nell'azienda.<br>**Salario**(Numerico): salario di un impiegato.<br>**DataScadenza**(Data): data di scadenza di un contratto per un impiegato a progetto.<br>**TipoContratto**(Stringa): indica se il contratto è a tempo indeterminato o "a progetto". |
| Promozione | Entità che tiene traccia degli scatti di carriera di un impiegato. | **CF**(Stringa): codice fiscale di un impiegato.<br>**DataPassaggio**(Data): data in cui avviene lo scatto di carriera di un impiegato.<br>**VecchiaCategoria**(Stringa): categoria di un impiegato al momento dello scatto di carriera.<br>**NuovaCategoria**(Stringa): categoria ottenuta dopo lo scatto di carriera.<br>**PromotoreDirigente**(Stringa): dirigente che permette a un senior di passare a dirigente. |
| Laboratorio | Entità che tiene traccia dei laboratori dell'azienda. | **Nome**(Stringa): nome di un laboratorio.<br>**Resp.Sci**(Stringa): responsabile scientifico di un laboratorio.<br>**Topic**(Stringa): indica di cosa si occupa un laboratorio.<br>**N.Afferenti**(Int): indica il numero di impiegati che lavorano a un laboratorio. |

| | | |
|---|---|---|
| Attrezzatura | Entità che tiene traccia delle attrezzature acquistate | **Seriale**(Stringa): identificativo di un'attrezzatura.<br>**Tipo**(Stringa): indica la tipologia di un'attrezzatura.<br>**Costo**(Numeric): costo di un'attrezzatura.<br>**NomeLab**(Stringa): nome del laboratorio a cui è associata un'attrezzatura.<br>**CUP**(Stringa): identificativo del progetto per cui è stata acquistata un'attrezzatura. |
| Progetto | Entità che tiene traccia dei progetti gestiti dall'azienda. | **CUP**(Stringa): identificativo di un progetto.<br>**Ref.Sci**(Stringa): referente scientifico di un progetto, necessariamente senior.<br>**Resp.**(Stringa): responsabile di un progetto, necessariamente dirigente.<br>**Nome**(Stringa): nome di un progetto.<br>**Budget**(Numeric): fondi a disposizione per un progetto. |

## 3.2 Dizionario delle Associazioni

| **Associazione** | **Tipologia** | **Descrizione** |
|---|---|---|
| Ottiene | Uno-a-Molti | Associazione che specifica che un impiegato può ottenere una o più promozioni nell'arco della carriera. |
| Utilizza | Molti-a-Molti | Un impiegato può lavorare in più laboratori e, allo stesso tempo, in un laboratorio possono lavorare più impiegati. |
| Possiede | Uno-a-Molti | Associa più attrezzature in un laboratorio e ogni attrezzatura in un solo laboratorio. |
| Lavora | Uno-a-Molti | A un progetto lavorano al più tre laboratori; un laboratorio lavora al massimo a un progetto. |
| Acquista | Uno-a-Molti | Per un progetto vengono acquistate più attrezzature; un'attrezzatura viene adoperata per un solo progetto. |

## 3.3 Dizionario dei Vincoli

| **Vincolo** | **Tipologia** | **Descrizione** |
|---|---|---|
| Di Responsabilità Scientifica | Inter-relazione | In un laboratorio il responsabile scientifico deve essere obbligatoriamente un impiegato senior. |
| Di Unicità | Intra-relazione | Il nome di un progetto deve essere unico nel sistema. |
| Di Referenza Scientifica | Inter-relazione | Per un progetto il referente scientifico deve essere obbligatoriamente un impiegato senior. |
| Di Responsabilità | Inter-relazione | Per un progetto il responsabile deve essere obbligatoriamente un dirigente. |
| Di Partizionamento | Inter-relazione | A un progetto possono lavorare al massimo tre laboratori. |
| Di Bilancio | Inter-relazione | Il 50% del budget di un progetto è destinato all'assunzione degli impiegati "a progetto", il restante all'acquisto dell'attrezzatura. |

# 4 Modello Logico

## 4.1 Traduzione di Entità e Associazioni

Una volta finita la ristrutturazione del modello concettuale, passiamo alla descrizione del modello logico. Per quanto riguarda le entità del modello ristrutturato segue una traduzione letterale. Per le associazioni, invece, c'è bisogno di un'ulteriore specifica:

- **Ottiene**: si tratta di una relazione *Uno-a-Molti parziale*. Si basa sulla presenza della chiave esterna all'interno della tabella Promozione che si riferisce alla chiave primaria della tabella Impiegato.
- **Utilizza**: si tratta di una relazione *Molti-a-Molti totale*. È stata risolta introducendo una classe di associazione che contiene due chiavi esterne: una riferita alla chiave primaria della tabella Impiegato, l'altra riferita alla chiave primaria della tabella Laboratorio.
- **Possiede**: si tratta di una relazione *Uno-a-Molti totale*. Si basa sulla presenza della chiave esterna NomeLab della tabella Attrezzatura che si riferisce alla chiave primaria della tabella Laboratorio.
- **Lavora**: si tratta di una relazione *Molti-a-Molti totale*. È stata risolta introducendo una classe di associazione che contiene due chiavi esterne: una riferita alla chiave primaria della tabella Laboratorio, l'altra alla chiave primaria della tabella Progetto.
- **Acquista**: si tratta di una relazione *Uno-a-Molti totale*. Si basa sulla presenza della chiave esterna CUP della tabella Attrezzatura che si riferisce alla chiave primaria della tabella Progetto.

## 4.2 Relazioni

***Legenda***: Chiave Primaria, Chiave Esterna, <Attributo Derivato>.

- **Impiegato**(CF, Nome, Cognome, DataNascita, DataAssunzione, <Eta>, Categoria, Salario, DataScadenza, TipoContratto);
- **Promozione**(CF, DataPassaggio, VecchiaCategoria, NuovaCategoria, PromotoreDirigente)
  - CF → Impiegato.CF
  - PromotoreDirigente → Impiegato.CF
- **Laboratorio**(Nome, Resp.Sci, Topic, <N.Afferenti>)
  - Resp.Sci → Impiegato.CF

- **Attrezzatura**(Seriale, Tipo, Costo, NomeLab, CUP)
  - NomeLab → Laboratorio.Nome
  - CUP → Progetto.CUP
- **Progetto**(CUP, Ref.Sci, Resp, Nome, Budget)
  - Ref.Sci → Impiegato.CF
  - Resp → Impiegato.CF
- **Utilizza**(CF, NomeLab)
  - CF → Impiegato.CF
  - NomeLab → Laboratorio.Nome
- **Lavora**(Laboratorio, CUP)
  - Laboratorio → Laboratorio.Nome
  - CUP → Progetto.CUP

# 5 Modello Fisico

## 5.1 Domini

### 5.1.1 Dominio Categoria

```
1    CREATE DOMAIN dominio_categoria AS varchar(20)
2    CONSTRAINT dominio_categoria_check CHECK (VALUE IN ('junior',
      'middle', 'senior', 'dirigente'));
```

Il dominio *dominio_categoria* indica la categoria che un impiegato può assumere all'interno dell'azienda.

### 5.1.2 Dominio Codice Fiscale

```
1    CREATE DOMAIN dominio_cf AS varchar(16)
2    CONSTRAINT dominio_cf_check CHECK (VALUE ~ '^[A-Za-z]{6}\d
     {2}[A-Za-z]\d{2}[A-Za-z]\d{3}[A-Za-z]$');
```

Il dominio *dominio_cf* sfrutta una RegEx per verificare il codice fiscale sia sintatticamente corretto.

### 5.1.3 Dominio Tipo Contratto

```
1    CREATE DOMAIN dominio_tipo_contratto AS varchar(20)
2    CONSTRAINT dominio_tipo_contratto_check CHECK (VALUE IN
3    ('indeterminato', 'progetto'));
```

Il dominio *dominio_tipo_contratto* indica i valori che può assumere il contratto di un impiegato.

## 5.2 Tabelle

### 5.2.1 Impiegato

```
1   CREATE TABLE IF NOT EXISTS impiegato (
2       cf              dominio_cf              NOT NULL,
3       nome            varchar(20)             NOT NULL,
4       cognome         varchar(20)             NOT NULL,
5       data_nascita    date                    NOT NULL,
6       data_assunzione date                    NOT NULL,
7       eta             integer                 NOT NULL,
8       categoria       dominio_categoria,
9       salario         numeric,
10      data_scadenza   date,
11      tipo_contratto  dominio_tipo_contratto NOT NULL,
12      CONSTRAINT impiegato_pk
13          PRIMARY KEY (cf),
14      CONSTRAINT impiegato_salario_check
15          CHECK (salario > 0)
16  );
```

**Nota**: il vincolo *impiegato_salario_check* serve per controllare che il salario inserito abbia un valore maggiore di 0. Questo perché è stato deciso che deve essere definito al momento di inserimento dell'impiegato e non successivamente.

### 5.2.2 Promozione

```
1   CREATE TABLE IF NOT EXISTS promozione (
2       cf                  dominio_cf        NOT NULL,
3       data_passaggio      date              NOT NULL,
4       vecchia_categoria   dominio_categoria NOT NULL,
5       nuova_categoria     dominio_categoria NOT NULL,
6       promotore_dirigente dominio_cf,
7       CONSTRAINT cf_fk
8           FOREIGN KEY (cf) REFERENCES impiegato,
9       CONSTRAINT promotore_dirigente_fk
10          FOREIGN KEY (promotore_dirigente) REFERENCES
11          impiegato
12  );
```

### 5.2.3 Laboratorio

```
1   CREATE TABLE IF NOT EXISTS laboratorio (
2       nome        varchar(20)       NOT NULL,
3       resp_sci    dominio_cf        NOT NULL,
4       topic       varchar(50)       NOT NULL,
5       n_afferenti integer DEFAULT 0 NOT NULL,
6   CONSTRAINT laboratorio_pk
7       PRIMARY KEY (nome),
8   CONSTRAINT respsci_nome_unique
9       UNIQUE (nome, resp_sci),
10  CONSTRAINT resp_sci_fk
11      FOREIGN KEY (resp_sci) REFERENCES impiegato,
12  CONSTRAINT n_afferenti_check
13      CHECK (n_afferenti >= 0)
14  );
```

**Nota**: il vincolo *n_afferenti_check* serve a controllare che il numero di afferenti di un laboratorio non sia mai minore di 0, tranne nel momento di creazione della tabella. Il vincolo *respsci_nome_unique* serve per fare in modo che un laboraotorio non abbia più responsabili scientifici e che, viceversa, un resposabile scientifico non si occupi di più laboratori contemporaneamente.

### 5.2.4 Progetto

```
1   CREATE TABLE IF NOT EXISTS progetto (
2       cup     varchar(15) NOT NULL,
3       ref_sci dominio_cf  NOT NULL,
4       resp    dominio_cf  NOT NULL,
5       nome    varchar(20) NOT NULL,
6       budget  numeric     NOT NULL,
7       CONSTRAINT progetto_pk
8           PRIMARY KEY (cup),
9       CONSTRAINT refsci_unique
10          UNIQUE (ref_sci),
11      CONSTRAINT resp_unique
12          UNIQUE (resp),
13      CONSTRAINT ref_sci_fk
14          FOREIGN KEY (ref_sci) REFERENCES impiegato,
15      CONSTRAINT resp_fk
16          FOREIGN KEY (resp) REFERENCES impiegato,
17      CONSTRAINT budget_check
18          CHECK (budget > 0)
19  );
```

**Nota**: il vincolo *budget_check* serve a controllare che il budget sia un valore maggiore di 0, in quanto è stato deciso che deve essere definito al momento di inserimento di un progetto e non successivamente. I vincoli *refsci_unique* e *resp_unique* servono per rendere unici nel sistema il referente scientifico e il responsabile di un progetto, come riportato nella richiesta del cliente.

### 5.2.5 Lavora

```
1   CREATE TABLE IF NOT EXISTS lavora (
2       laboratorio varchar(20) NOT NULL,
3       cup         varchar(15) NOT NULL,
4       CONSTRAINT laboratorio_unique
5           UNIQUE (laboratorio),
6       CONSTRAINT laboratorio_fk
7           FOREIGN KEY (laboratorio) REFERENCES laboratorio
8               ON DELETE CASCADE,
9       CONSTRAINT cup_fk
10          FOREIGN KEY (cup) REFERENCES progetto
11              ON DELETE CASCADE
12  );
```

**Nota**: in *cup_fk*, il vincolo di integrità referenziale `ON DELETE CASCADE` specifica che, alla rimozione di un CUP dalla tabella Progetto, verranno rimosse tutte le tuple corrispondenti nella tabella in analisi. È presente anche in *laboratorio_fk* in quanto, nel caso in cui venga rimosso un laboratorio, verranno eliminate tutte le tuple nella tabella in questione. Non vengono fatti controlli sul progetto associato perché è stato deciso che questo può esistere anche senza un laboratorio. Il vincolo *laboratorio_unique* serve per far in modo che non ci siano più occorrenze di un laboratorio nella tabella in analisi. È stato aggiunto perché è stato deciso che a un laboratorio può essere affidato un solo progetto.

### 5.2.6 Utilizza

```
1   CREATE TABLE IF NOT EXISTS utilizza (
2       nome_lab varchar(20) NOT NULL,
3       cf       dominio_cf  NOT NULL,
4       CONSTRAINT nome_lab_fk
5           FOREIGN KEY (nome_lab) REFERENCES laboratorio
6               ON DELETE CASCADE,
7       CONSTRAINT cf_fk
8           FOREIGN KEY (cf) REFERENCES impiegato
9   );
```

**Nota**: in *nome_lab_fk*, il vincolo di integrità referenziale `ON DELETE CASCADE` specifica che, alla rimozione di un nome nella tabella Laboratorio, verranno rimosse tutte le tuple corrispondenti nella tabella in analisi.

### 5.2.7 Attrezzatura

```
1   CREATE TABLE IF NOT EXISTS attrezzatura (
2       seriale  varchar(10) NOT NULL,
3       tipo     varchar(20) NOT NULL,
4       costo    numeric     NOT NULL,
5       nome_lab varchar(20),
6       cup      varchar(15),
7       CONSTRAINT attrezzatura_pk
8           PRIMARY KEY (seriale),
9       CONSTRAINT seriale_cup_unique
10          UNIQUE (seriale, cup),
11      CONSTRAINT nome_lab_fk
12          FOREIGN KEY (nome_lab) REFERENCES laboratorio,
13      CONSTRAINT cup_fk
14          FOREIGN KEY (cup) REFERENCES progetto,
15      CONSTRAINT costo_check
16          CHECK (costo > 0)
17  );
```

**Nota**: il vincolo *costo_check* implica che il costo debba sempre avere un valore maggiore di 0, in quanto è stato deciso che deve essere definito al momento di inserimento di un'attrezzatura e non successivamente. Il vincolo *seriale_cup_unique* serve per fare in modo che un'attrezzatura venga acquistata per un solo progetto.

## 5.3 Viste

### 5.3.1 Lista Attrezzature Progetto

```
1   CREATE OR REPLACE VIEW lista_attrezzature_progetto(cup,
    costo_totale) AS
2       SELECT cup,
3       (SELECT sum(a.costo) AS sum
4       FROM attrezzatura a
5       WHERE a.cup = p.cup) AS costo_totale
6       FROM progetto p;
```

Questa vista mostra il budget speso da un progetto per l'acquisto delle varie attrezzature. Viene utilizzata nella funzione *f_check_acquista_attrezzatura* per il calcolo del badget residuo di un progetto.

### 5.3.2 Lista Impiegati Progetto

```
1   CREATE OR REPLACE VIEW lista_impiegati_progetto(cup,
    stipendi_totali) AS
2       SELECT cup,
3       (SELECT sum(i.salario) AS sum
4       FROM impiegato i, utilizza u, lavora l
5       WHERE i.cf = u.cf
6         AND u.nome_lab = l.laboratorio
7         AND l.cup = p.cup
8         AND i.tipo_contratto = 'progetto') AS stipendi_totali
9       FROM progetto p;
```

Questa vista mostra il budget speso da un progetto per l'assunzione dei dipendenti con contratto "a progetto". Viene utilizzata nella procedura *add_impiegato_progetto* per calcolare se è c'è badget sufficiente per la suddetta operazione.

### 5.3.3 Afferenti Progetto

```
1    CREATE OR REPLACE afferenti_progetto(cup, n_afferenti) AS
2    SELECT lv.cup, sum(lb.n_afferenti) AS n_afferenti
3    FROM lavora lv
4         JOIN laboratorio lb ON lv.laboratorio = lb.nome
5    GROUP BY lv.cup;
```

Questa vista mostra il numero di afferenti per ogni progetto, andandoli ad estrapolare dai vari laboratori che lavorano allo stesso.

### 5.3.4 Tabulato Dipendenti

```
1    CREATE OR REPLACE VIEW tabulato_dipendenti(categoria,
     numero_dipendenti) AS
2    SELECT categoria, count(cf) AS numero_dipendenti
3    FROM impiegato
4    WHERE categoria IS NOT NULL
5    GROUP BY categoria;
```

Questa vista mostra il numero di dipendenti raggruppati per categoria ("Junior", "Middle", "Senior", "Dirigente").

### 5.3.5 Tabulato Contratti

```
1    CREATE OR REPLACE VIEW tabulato_contratti(tipo_contratto,
     numero_dipendenti) AS
2    SELECT tipo_contratto, count(cf) AS numero_dipendenti
3    FROM impiegato
4    GROUP BY tipo_contratto;
```

Questa vista mostra il numero di dipendenti raggruppati per la tipologia di contratto ("Indeterminato" o "A Progetto").

## 5.4 Procedure e Funzioni

### 5.4.1 add_progetto

```
1   CREATE OR REPLACE PROCEDURE add_progetto(IN _cup dominio_cf,
    IN _ref_sci dominio_cf, IN _resp dominio_cf, IN _nome varchar
    )
2   LANGUAGE plpgsql
3   AS
4   $$
5       DECLARE
6           _budget integer=FLOOR(RANDOM()*(100000 - 50000 + 1))
7           + 50000;
8       BEGIN
9           IF _ref_sci in (SELECT resp_sci FROM laboratorio)
10          THEN
11              RAISE EXCEPTION 'Il seguente impiegato si occupa
12              giá di un laboratorio.';
13          END IF;
14
15          INSERT INTO progetto (cup, ref_sci, resp,
16          nome, budget)
17          VALUES (_cup, _ref_sci, _resp, _nome, _budget);
18      END;
19  $$;
```

La procedura in questione si occupa di inserire un nuovo progetto. In particolare, viene generato, per comodità, randomicamente il badget e viene fatto un controllo sul responsabile scientifico, in quanto è stato deciso che non si può occupare allo stesso tempo anche di un laboratorio. In caso di violazione della condizione, la procedura lancia un'eccezione.

### 5.4.2 add_attrezzatura

```
1   CREATE OR REPLACE PROCEDURE add_attrezzatura(IN _seriale
    varchar, IN _tipo varchar)
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       _costo numeric = FLOOR(RANDOM()*(2000-100+1))+100;
7   BEGIN
8       INSERT INTO attrezzatura(seriale, tipo, costo)
9       VALUES (_seriale, _tipo, _costo);
10  END;
11  $$;
```

La procedura in questione si occupa di inserire una nuova attrezzatura. L'unica operazione aggiuntiva è la randomica generazione del costo della stessa per comodità.

### 5.4.3 add_laboratorio

```
1  CREATE OR REPLACE PROCEDURE add_laboratorio(IN _nome varchar,
2  IN _resp_sci dominio_cf, IN _topic varchar)
3  LANGUAGE plpgsql
4  AS
5  $$
6  BEGIN
7      IF _resp_sci in (SELECT ref_sci FROM progetto) THEN
8          RAISE EXCEPTION 'Il seguente impiegato si occupa
9          giá di un progetto.';
10     END IF;
11
12     INSERT INTO laboratorio(nome, resp_sci, topic)
13     VALUES (_nome, _resp_sci, _topic);
14 END;
15 $$;
```

La procedura in questione si occupa di inserire un nuovo laboratorio. In particolare, viene effettuato un controllo sul referente scientifico, in quanto è stato deciso che non si può occupare allo stesso tempo anche di un progetto. In caso di violazione della condizione, la procedura lancia un'eccezione.

### 5.4.4 generate_random_salary

```
1  CREATE OR REPLACE FUNCTION generate_random_salary(_categoria
   varchar) RETURNS numeric
2  LANGUAGE plpgsql
3  AS
4  $$
5  BEGIN
6      IF _categoria = 'junior' THEN
7          RETURN FLOOR(random()*(30000 - 20000 + 1)) + 20000;
8      ELSIF _categoria = 'middle' THEN
9          RETURN FLOOR(random()*(50000 - 31000 + 1)) + 31000;
10     ELSIF _categoria = 'senior' THEN
11         RETURN FLOOR(random()*(70000 - 51000 + 1)) + 51000;
12     ELSE
13         RETURN FLOOR(random()*(100000 - 71000 + 1)) + 71000;
14     END IF;
15 END;
16 $$;
```

La seguente funzione si occupa della generazione randomica del salario di un impiegato in base alla categoria a cui appartiene.

### 5.4.5 add_impiegato_progetto

```
1   CREATE OR REPLACE PROCEDURE add_impiegato_progetto(IN _cf
    dominio_cf, IN _nome varchar, IN _cognome varchar, IN
    _data_nascita date, IN _data_assunzione date, IN
    _data_scadenza date, IN _cup varchar)
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       _laboratorio         varchar(20) = (SELECT laboratorio
7                                           FROM lavora
8                                           WHERE cup = _cup
9                                           LIMIT 1);
10      _stipendio_rimanente numeric     = (SELECT p.budget / 2
11                                          FROM progetto p
12                                          WHERE p.cup = _cup) -
13                                      (SELECT stipendi_totali
14                                      FROM
15                                      lista_impiegati_progetto
16                                      lista
17                                      WHERE lista.cup = _cup);
18      _salario             numeric     = FLOOR(random() *
19                                         (15000 - 10000 + 1)) +
20                                         10000;
21  BEGIN
22      IF date_part('year', current_date) -
23      date_part('year', _data_nascita) < 18 THEN
24          RAISE EXCEPTION 'Bisogna avere almeno 18 anni per
25          essere assunti.';
26      END IF;
27
28
29      IF _data_scadenza IS NULL OR _data_scadenza <
30      _data_assunzione THEN
31          RAISE EXCEPTION 'Mancanza della data di scadenza o
32          errata.';
33      END IF;
34
35      IF (_stipendio_rimanente < _salario) THEN
36          RAISE EXCEPTION 'Mancanza di fondi, impossibile
37          assumere.';
38      END IF;
39
40      INSERT INTO impiegato (cf, nome, cognome, data_nascita,
41      data_assunzione, eta, categoria, salario,
42      data_scadenza, tipo_contratto)
43      VALUES (_cf, _nome, _cognome, _data_nascita,
44      _data_assunzione, date_part('YEAR', current_date) -
45      date_part('YEAR', _data_nascita), null, _salario,
46      _data_scadenza, 'progetto');
47
48      INSERT INTO utilizzacontratto
49      VAUES (_laboratorio, _cf);
50  END;
51  $$;
```

La seguente procedura si occupa di inserire un impiegato con contratto "a progetto". Vengono effettuati diversi controlli: uno sull'età, in quanto deve essere maggiorenne; uno sulla data di scandenza del contratto e uno sul salario, in quanto può essere assunto solo nel caso in cui il badget del progetto a cui è associato è sufficiente. Alla violazione di almeno una delle condizioni, la procedura lancia un'eccezione. Viene, infine, fatto anche un inserimento nella tabella Utilizza.

### 5.4.6 add_impiegato

```
1  CREATE OR REPLACE PROCEDURE add_impiegato(IN _cf dominio_cf,
   IN _nome varchar, IN _cognome varchar, IN _data_nascita date,
2  IN _data_assunzione date, IN _categoria dominio_categoria)
3  LANGUAGE plpgsql
4  AS
5  $$
6  DECLARE
7      _salario numeric;
8  BEGIN
9      IF date_part('year', current_date) - date_part('year',
10     _data_nascita) < 18 THEN
11         RAISE EXCEPTION 'Bisogna avere almeno 18 anni per
12         essere assunti.';
13     END IF;
14
15     IF _categoria IS NULL THEN
16         _categoria = 'junior';
17     END IF;
18
19     _salario = generate_random_salary(_categoria);
20
21     INSERT INTO impiegato(CF, nome, cognome, data_nascita,
22     data_assunzione, eta, categoria, salario, data_scadenza,
23     tipo_contratto)
24     VALUES (_cf, _nome, _cognome, _data_nascita,
25     _data_assunzione, date_part('YEAR', current_date) -
26     date_part('YEAR', _data_nascita), _categoria,
27     _salario, null, 'indeterminato');
28 END;
29 $$;
```

La seguente procedura si occupa di inserire un impiegato con contratto a tempo indeterminato. Vengono fatti due controlli: uno sull'età, in quanto deve essere maggiorenne; l'altro sulla categoria, in quanto, se non viene inserita manualmente, deve essere settata automaticamente a 'junior'. Il salario viene generato randomicamente tramite la funzione *generate_random_salary*.

### 5.4.7 delete_impiegato

```
1     CREATE OR REPLACE PROCEDURE delete_impiegato(IN _cf
      dominio_cf)
2     LANGUAGE plpgsql
3     AS
4     $$
5     BEGIN
6         DELETE FROM impiegato i WHERE i.cf = _cf;
7     END;
8     $$;
```

La seguente procedura si occupa della rimozione di un impiegato. Vale sia per quelli con contratto a tempo indeterminato sia per quelli con contratto "a progetto".

### 5.4.8 substitute_impiegato

```
1     CREATE OR REPLACE PROCEDURE substitute_impiegato(IN _old_cf
      dominio_cf, IN _new_cf dominio_cf)
2     LANGUAGE plpgsql
3     AS
4     $$
5     BEGIN
6         UPDATE laboratorio
7         SET resp_sci = _new_cf
8         WHERE resp_sci = _old_cf;
9
10        UPDATE progetto
11        SET ref_sci = _new_cf
12        WHERE ref_sci = _old_cf;
13
14        UPDATE progetto
15        SET resp = _new_cf
16        WHERE resp = _old_cf;
17    END;
18    $$;
```

La seguente procedura si occupa di sostituire o un responsabile scientifico nella tabella Laboratorio, o un referente scientifico o un responsabile nella tabella Progetto. È stata implementata per garantire la condizione presente nella funzione *f_check_delete_impiegato* associata al trigger *t_check_delete_impiegato*.

### 5.4.9 add_promozione

```
1   CREATE OR REPLACE PROCEDURE add_promozione(IN _cf dominio_cf,
    IN _promotore_dirigente dominio_cf)
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       _vecchia_categoria varchar(20) = (SELECT categoria
7                                         FROM impiegato
8                                         WHERE _CF =
9                                         impiegato.cf);
10      _nuova_categoria   varchar(20);
11
12  BEGIN
13      IF _vecchia_categoria = 'junior' THEN
14          _nuova_categoria = 'middle';
15      ELSIF _vecchia_categoria = 'middle' THEN
16          _nuova_categoria = 'senior';
17      ELSE
18          _nuova_categoria = 'dirigente';
19      END IF;
20
21      INSERT INTO promozione
22      VALUES (_CF, CURRENT_DATE, _vecchia_categoria,
23      _nuova_categoria, _promotore_dirigente);
24
25      UPDATE impiegato
26      SET categoria = _nuova_categoria
27      WHERE cf = _cf;
28  END;
29  $$;
30
```

La seguente procedura si occupa di inserire una promozione di un impiegato. In base alla vecchia categoria, viene settata in automatico la nuova. Oltre all'inserimento nella tabella Promozione, viene effettauta anche una modifica della categoria dell'impiegato nella tabella Impiegato.

### 5.4.10 add_utilizza

```
1   CREATE OR REPLACE PROCEDURE add_utilizza(IN _cf dominio_cf,
    IN _nomelab varchar)
2   LANGUAGE plpgsql
3   AS
4   $$
5   BEGIN
6       INSERT INTO utilizza
7       VALUES (_nomeLab, _cf);
8
9   END;
10  $$;
```

La seguente procedura si occupa dell'inserimento di una nuova tupla nella tabella Utilizza.

### 5.4.11 add_lavora

```
1    CREATE OR REPLACE PROCEDURE add_lavora(IN _laboratorio
     varchar, IN _cup varchar)
2    LANGUAGE plpgsql
3    AS
4    $$
5    BEGIN
6        INSERT INTO lavora
7        VALUES (_laboratorio, _cup);
8    END;
9    $$;
```

La seguente procedura si occupa di inserire una nuova tupla nella tabella Lavora.

### 5.4.12 acquista_attrezzatura

```
1    CREATE OR REPLACE PROCEDURE acquista_attrezzatura(IN _seriale
      varchar, IN _nome_lab varchar)
2    LANGUAGE plpgsql
3    AS
4    $$
5    BEGIN
6        UPDATE attrezzatura
7        SET nome_lab = _nome_lab,
8        cup = (SELECT cup
9               FROM lavora
10              WHERE laboratorio = _nome_lab)
11       WHERE seriale = _seriale;
12   END;
13   $$;
```

La seguente procedura si occupa dell'acquisto di un'attrezzatura per un determinato progetto.

## 5.5 Triggers

### 5.5.1 check_cat_prog

```
1   CREATE OR REPLACE FUNCTION f_check_cat_prog() RETURNS trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       cat_ref_sci varchar(20) = (SELECT categoria
7                                  FROM impiegato
8                                  WHERE cf = new.ref_sci);
9
10      cat_resp varchar(20) = (SELECT categoria
11                              FROM impiegato
12                              WHERE cf = new.resp);
13  BEGIN
14      IF cat_ref_sci <> 'senior' or
15         cat_resp <> 'dirigente' THEN
16          RAISE EXCEPTION 'Categoria del Referente Scientifico
17          e/o del Dirigente non validi.';
18      END IF;
19
20      RETURN new;
21  END;
22  $$;
23
24  CREATE TRIGGER t_check_cat_prog
25      BEFORE INSERT OR UPDATE
26      ON progetto
27      FOR EACH ROW
28  EXECUTE PROCEDURE f_check_cat_prog();
```

Il trigger in questione viene attivato prima di un inserimento o di una modifica sulla tabella Progetto. Il compito della funzione associata è quello di verificare che il *Referente Scientifico* e il *Responsabile* siano della categoria opportuna (rispettivamente Senior e Dirigente), lanciando un eccezione in caso contrario.

### 5.5.2 check_cat_lab

```
1   CREATE OR REPLACE FUNCTION f_check_cat_lab() RETURNS trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       cat_resp_sci varchar(20) = (SELECT categoria
7                                   FROM impiegato
8                                   WHERE cf = new.resp_sci);
9   BEGIN
10      IF cat_resp_sci <> 'senior' THEN
11          RAISE EXCEPTION 'Categoria del Responsabile
12          Scientifico non valida.';
13      END IF;
14
15      RETURN new;
16  END;
17  $$;
18
19  CREATE TRIGGER t_check_cat_lab
20      BEFORE INSERT OR UPDATE
21      ON laboratorio
22      FOR EACH ROW
23  EXECUTE PROCEDURE f_check_cat_lab();
```

Il trigger in questione viene attivato prima di un inserimento o di una modifica sulla tabella Laboratorio. Il compito della funzione associata è quello di verificare che il *Responsabile Scientifico* sia della categoria opportuna (Senior), lanciando un'eccezione in caso contrario.

### 5.5.3 check_acquista_attrezzatura

```
1   CREATE OR REPLACE FUNCTION f_check_acquista_attrezzatura()
    RETURNS trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6   _budgetResiduo numeric = (SELECT p.budget / 2
7                             FROM progetto p
8                             WHERE p.cup = new.cup) -
9                            (SELECT sum(costo_totale)
10                            FROM lista_attrezzature_progetto
11                            WHERE cup = new.cup);
12  _prezzo        numeric = (SELECT a.costo
13                            FROM attrezzatura a
14                            WHERE new.seriale = a.seriale);
15  BEGIN
16      IF old.cup IS NOT NULL OR old.nome_lab IS NOT NULL THEN
17          RAISE EXCEPTION 'Attrezzatura giá assegnata a un
18          laboratorio/progetto';
19      END IF;
20      IF (_prezzo > _budgetResiduo) THEN
21          RAISE EXCEPTION 'Budget insufficiente';
22      END IF;
```

```
23
24          RETURN new;
25      END;
26      $$;
27
28      CREATE TRIGGER t_check_acquista_attrezzatura
29          BEFORE UPDATE
30          ON attrezzatura
31          FOR EACH ROW
32      EXECUTE PROCEDURE f_check_acquista_attrezzatura();
```

Il trigger in questione viene attivato prima di una modifica sulla tabella Attrezzatura. Il compito della funzione associata è quello di verificare che il *CUP* e il *NomeLab* siano stati inseriti correttamente, oltre a verificare che il progetto associato abbia abbastanza budget residuo. Lancia un'eccezione in caso di errore.

### 5.5.4 check_delete_impiegato

```
1       CREATE OR REPLACE FUNCTION f_check_delete_impiegato()
2       RETURNS trigger
3       LANGUAGE plpgsql
4       AS
5       $$
6       BEGIN
7           IF old.cf IN (SELECT resp_sci FROM laboratorio) OR
8           old.cf IN (SELECT ref_sci FROM progetto) OR
9           old.cf IN (SELECT resp FROM progetto) THEN
10              RAISE EXCEPTION 'Impossibile licenziare. Trova
11              prima un sostituto.';
12          END IF;
13
14          RETURN old;
15      END;
16      $$;
17
18      CREATE TRIGGER t_check_delete_impiegato
19          BEFORE DELETE
20          ON impiegato
21          FOR EACH ROW
22      EXECUTE PROCEDURE f_check_delete_impiegato();
```

Il trigger in questione viene attivato prima dell'eliminazione sulla tabella Impiegato. Il compito della funzione associata è quello di verificare che l'impiegato in questione non sia un responsabile scientifico di un laboratorio, il referente scientifico o responsabile di un progetto. In caso affermativo, la funzione lancia un'eccezione.

### 5.5.5 check_add_promozione

```
1   CREATE OR REPLACE FUNCTION f_check_add_promozione()
2   RETURNS trigger
3   LANGUAGE plpgsql
4   AS
5   $$
6   DECLARE
7       _anni_di_servizio integer = date_part('year',
8       current_date) - (SELECT date_part('year',data_assunzione)
9                        FROM impiegato
10                       WHERE cf = new.cf);
11      _cat_prom_dir dominio_categoria = (SELECT categoria
12                                FROM impiegato
13                                WHERE cf =
14                                new.promotore_dirigente);
15  BEGIN
16      IF new.nuova_categoria = 'middle' AND
17      _anni_di_servizio < 3 THEN
18          RAISE EXCEPTION 'Questo impiegato non ha abbastanza
19          esperienza';
20      ELSIF new.nuova_categoria = 'senior' AND
21      _anni_di_servizio < 7 THEN
22          RAISE EXCEPTION 'Questo impiegato non ha abbastanza
23          esperienza';
24      ELSIF new.nuova_categoria = 'dirigente' AND
25      (new.promotore_dirigente IS NULL OR _cat_prom_dir <>
26      'dirigente') THEN
27          RAISE EXCEPTION 'Promotore dirigente mancante o
28          errato';
29      END IF;
30
31      RETURN new;
32  END;
33  $$;
34
35  CREATE TRIGGER t_check_add_promozione
36      BEFORE insert
37      ON promozione
38      FOR EACH ROW
39  EXECUTE FUNCTION f_check_add_promozione();
```

Il trigger in questione viene attivato prima di un inserimento sulla tabella Promozione. Il compito della funzione associata è quello di verificare che gli anni di servizio di un impiegato siano coerenti con la promozione. Viene inoltre verificato se è presente il "promotore dirigente" per la promozione da Senior a Dirigente. Nel caso in cui una di queste due condizioni non si avveri viene lanciata un'eccezione.

### 5.5.6 check_add_utilizza

```
1   CREATE OR REPLACE FUNCTION f_check_add_utilizza() RETURNS
    trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       _laboratorio varchar(20) = (SELECT nome_lab
7                                   FROM utilizza
8                                   WHERE cf = new.cf
9                                   LIMIT 1);
10      _progetto varchar(15) = (SELECT cup
11                               FROM lavora
12                               WHERE _laboratorio = laboratorio);
13  BEGIN
14      IF (_progetto <> (SELECT cup FROM lavora WHERE
15      new.nome_lab = laboratorio)) THEN
16          RAISE EXCEPTION 'Questo impiegato si occupa giá di
17          un altro progetto.';
18      END IF;
19
20      RETURN new;
21  END;
22  $$;
23
24  CREATE TRIGGER t_check_add_utilizza
25      BEFORE INSERT
26      ON utilizza
27      FOR EACH ROW
28  EXECUTE PROCEDURE f_check_add_utilizza();
```

Il trigger in questione viene attivato prima di un inserimento sulla tabella Utilizza. Il compito della funzione associata è quello di verificare che l'impiegato lavori ad un solo progetto contemporaneamente, lanciando un'eccezione in caso contrario.

### 5.5.7 check_add_impiegato

```
1   CREATE OR REPLACE FUNCTION f_check_add_impiegato() RETURNS
    trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   DECLARE
6       _anni_di_servizio integer = date_part('year',
7       current_date) - date_part('year', new.data_assunzione);
8   BEGIN
9       ALTER TABLE promozione
10          DISABLE TRIGGER t_check_add_promozione;
11
12      IF new.categoria = 'middle' AND _anni_di_servizio >= 4
13      THEN
14          INSERT INTO promozione (cf, data_passaggio,
15          vecchia_categoria, nuova_categoria,
16          promotore_dirigente)
17          VALUES (new.cf, new.data_assunzione + interval
18          '4 years', new.categoria, 'senior', null);
19
20          UPDATE impiegato
21          SET categoria = 'senior', salario =
22          generate_random_salary('senior')
23          WHERE cf = new.cf;
24      END IF;
25
26      IF new.categoria = 'junior' THEN
27          IF _anni_di_servizio >= 7 THEN
28              INSERT INTO promozione (cf, data_passaggio,
29              vecchia_categoria, nuova_categoria,
30              promotore_dirigente)
31              VALUES (new.cf, new.data_assunzione + interval
32              '3 years', new.categoria, 'middle', null),
33              (new.cf, new.data_assunzione + interval
34              '7 years', 'middle', 'senior', null);
35
36              UPDATE impiegato
37              SET categoria = 'senior', salario =
38              generate_random_salary('senior')
39              WHERE cf = new.cf;
40
41          ELSIF _anni_di_servizio >= 3 THEN
42              INSERT INTO promozione (cf, data_passaggio,
43              vecchia_categoria, nuova_categoria,
44              promotore_dirigente)
45              VALUES (new.cf, new.data_assunzione + interval
46              '3 years', new.categoria, 'middle', null);
47
48              UPDATE impiegato
49              SET categoria = 'middle',
50              salario   = generate_random_salary('middle')
51              WHERE cf = new.cf;
52          END IF;
53      END IF;
54
```

```
55          ALTER TABLE promozione
56              ENABLE TRIGGER t_check_add_promozione;
57
58          RETURN new;
59      END;
60      $$;
61
62      CREATE TRIGGER t_check_add_impiegato
63          AFTER INSERT
64          IN impiegato
65          FOR EACH ROW
66      EXECUTE procedure f_check_add_impiegato();
```

Il trigger in questione viene attivato prima di un inserimento sulla tabella Impiegato. Per prima cosa la funzione associata disabilita il trigger *check_add_promozione* per evitare eventuali conflitti con il corpo della funzione. Successivamente controlla se l'impiegato appena inserito abbia abbastanza anni di servizio per effettuare dei salti di carriera in base alla categoria di partenza (è possibile inserire un impiegato in una specifica categoria come da funzione *add_impiegato*).

### 5.5.8 check_add_impiegato_progetto

```
1       CREATE OR REPLACE FUNCTION f_check_add_impiegato_progetto()
        RETURNS trigger
2       LANGUAGE plpgsql
3       AS
4       $$
5       DECLARE
6           _cup_impiegato varchar(20) = (SELECT l.cup
7                                         FROM lavora l
8                                         JOIN utilizza u ON
9                                         u.nome_lab = l.lab1
10                                        WHERE u.cf = new.cf);
11          _budgetResiduo numeric     = (SELECT p.budget / 2 -
12                                        li.stipendi_totali
13                                        FROM progetto p,
14                                        lista_impiegati_progetto li
15                                        WHERE p.cup = li.cup
16                                        AND _cup_impiegato=p.cup);
17      BEGIN
18          IF (_budgetResiduo < new.salario) THEN
19              DELETE
20              FROM utilizza
21              WHERE new.cf = cf;
22              RETURN old;
23          END IF;
24          RETURN new;
25      END;
26      $$;
27
28
29
30
31
```

```
32      CREATE TRIGGER t_check_add_impiegato_progetto
33          BEFORE INSERT
34          ON impiegato
35          FOR EACH ROW
36      EXECUTE PROCEDURE f_check_add_impiegato_progetto();
```

Il trigger in questione viene attivato prima di un inserimento sulla tabella Impiegato. Il compito della funzione associata è quello di verificare che il progetto abbia abbastanza fondi per assumere l'impiegato in questione, lanciando un'eccezione in caso contrario.

### 5.5.9 check_add_lavora

```
1       CREATE OR REPLACE FUNCTION f_check_add_lavora() RETURNS
        trigger
2       LANGUAGE plpgsql
3       AS
4       $$
5       BEGIN
6           IF (SELECT count(*) FROM lavora WHERE new.cup = cup) = 3
        THEN
7               RAISE EXCEPTION 'Numero massimo di laboratori
8               raggiunto per questo progetto.';
9           END IF;
10          RETURN new;
11      END;
12      $$;
13
14      CREATE TRIGGER t_check_add_lavora
15          BEFORE INSERT
16          ON lavora
17          FOR EACH ROW
18      EXECUTE PROCEDURE f_check_add_lavora();
```

Il trigger in questione viene attivato prima di un inserimento sulla tabella Lavora. Il compito della funzione associata è quello di verificare che il numero di laboratori che lavorano ad un progetto non superi il limite massimo di 3. In caso di superamento viene lanciata un'eccezione.

### 5.5.10 update_lab

```
1   CREATE OR REPLACE FUNCTION f_update_lab() RETURNS trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   BEGIN
6       UPDATE laboratorio
7       SET n_afferenti = n_afferenti - 1
8       WHERE nome = old.nome_lab;
9
10      UPDATE laboratorio
11      SET n_afferenti = n_afferenti + 1
12      WHERE nome = new.nome_lab;
13
14      RETURN new;
15  END;
16  $$;
17
18  CREATE TRIGGER t_update_lab
19      AFTER UPDATE OF nome_lab
20      ON utilizza
21      FOR EACH ROW
22  EXECUTE PROCEDURE f_update_lab();
```

Il trigger in questione viene attivato dopo un aggiornamento sul campo *nome_lab* della tabella Utilizza. Il compito della funzione associata è quello di aggiornare il valore di *n_afferenti* all'interno della tabella Laboratorio nel momento in cui viene cambiato il laboratorio associato all'impiegato.

### 5.5.11 add_afferente

```
1   CREATE OR REPLACE FUNCTION f_add_afferente() RETURNS trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5       BEGIN
6           UPDATE laboratorio
7           SET n_afferenti = n_afferenti + 1
8           WHERE nome = new.nome_lab;
9
10          RETURN new;
11      END;
12  $$;
13
14  CREATE TRIGGER t_add_afferente
15      AFTER INSERT
16      ON utilizza
17      FOR EACH ROW
18  EXECUTE PROCEDURE f_add_afferente();
```

Il trigger in questione viene attivato dopo un inserimento nella tabella Utilizza. Il compito della funzione associata è quello di incrementare di 1 il valore di *n_affereneti*, nella tabella Laboratorio, del laboratorio in cui è stato fatto l'inserimento.

### 5.5.12 remove_afferente

```
1   CREATE OR REPLACE FUNCTION f_remove_afferente() RETURNS
    trigger
2   LANGUAGE plpgsql
3   AS
4   $$
5   BEGIN
6       UPDATE laboratorio
7       SET n_afferenti = n_afferenti - 1
8       WHERE nome = old.nome_lab;
9
10      RETURN new;
11  END;
12  $$;
13
14  CREATE TRIGGER t_remove_afferente
15      AFTER DELETE
16      ON utilizza
17      FOR EACH ROW
18  EXECUTE PROCEDURE f_remove_afferente();
```

Il trigger in questione viene attivato dopo la rimozione di un elemento dalla tabella Utilizza. Il compito della funzione associata è quello di decrementare di 1 il valore di *n_afferenti*, nella tabella Laboratorio, del laboratorio in cui è stata fatta la rimozione.