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

Roberto Ingenito Simone Ingenito Lorenzo Sequino

6 gennaio 2023

# Indice

**1 Progettazione concettuale** **3**
1.1 Analisi dei requisiti . . . 3
1.2 Schema concettuale . . . 4
1.3 Dizionario delle entità . . . 5
1.4 Dizionario delle associazioni . . . 7

**2 Ristrutturazione del modello concettuale** **8**
2.1 Analisi delle ridondanze . . . 8
2.2 Eliminazione degli attributi multivalore . . . 8
2.3 Eliminazione degli attributi composti . . . 8
2.4 Analisi delle generalizzazioni . . . 8
2.5 Identificazioni chiavi primarie . . . 8
2.6 Schema ristrutturato UML . . . 9
2.7 Schema ristrutturato ER . . . 9

**3 Traduzione al modello logico** **10**
3.1 Mapping associazioni . . . 10
3.1.1 Associazioni 1-N . . . 10
3.1.2 Associazioni N-N . . . 10
3.2 Modello logico . . . 11

**4 Progettazione fisica** **12**
4.1 Creazione del database . . . 12
4.2 Creazione dello schema . . . 12
4.3 Attivazione estensioni . . . 12
4.4 Gestione ruoli e permessi . . . 12
4.5 Creazione domini . . . 12
4.6 Creazione tabelle . . . 13
4.7 Trigger e trigger function . . . 17
4.8 Procedure e funzioni . . . 17
4.9 Dizionario dei vincoli . . . 17
4.9.1 Vincoli intra-relazionali . . . 17
4.9.2 Vincoli inter-relazionai . . . 21

# 1 Progettazione concettuale

## 1.1 Analisi dei requisiti

La base dati si deve occupare della gestione del personale di un'azienda:
la figura centrale di tutto lo schema è l'impiegato.

La prima classificazione di un generico impiegato viene fatta tra:

- impiegato assunto regolarmente
- impiegato assunto esclusivamente per lavorare ad un progetto

Gli impiegati assunti regolarmente, vengono assunti con un determinato ruolo e stipendio, inoltre hanno diverse specializzazioni in base sia al merito sia agli anni trascorsi all'interno dell'azienda.

Dal momento in cui viene assunto, l'impiegato diventa automaticamente un junior. Trascorsi 3 anni diventa middle, trascorsi altri 4 diventa senior. Inoltre, qualunque siano gli anni trascorsi all'interno dell'azienda, può essere promosso e diventare manager.

Si tiene traccia di tutti gli scatti di carriera all'interno dell'entità *career_log* dove sono memorizzati:

- ruolo precedente
- ruolo successivo
- data dello scatto

L'azienda è divisa in varie sedi dove sono presenti diversi laboratori.

Ogni impiegato afferisce ad un unico laboratorio.
Un impiegato può aver anche lavorato in più laboratori ma in periodi diversi.
Al momento dell'assunzione l'impiegato non ha ancora una collocazione ma viene stabilita successivamente.
Ad un impiegato senior può essere assegnata la gestione di laboratori e/o di progetti.

Ad ogni laboratorio è assegnato un manager scientifico che è un impiegato senior.

L'amministratore, a cui è affidata la gestione del database, si occupa di:

- inserire gli impiegati assunti
- monitorare gli scatti di carriera
- promuovere gli impiegati meritevoli a manager
- creare progetti e affidare la supervisione ad un manager e la gestione ad un referente scientifico

Per ogni progetto viene stabilita una data di inizio, una *deadline* per la conclusione del progetto e dei fondi.

Inoltre, si tiene traccia dell'effettiva data in cui il progetto è terminato *end_date*; può accadere che il progetto si protragga oltre la deadline

Ad un progetto possono prendere parte massimo 3 laboratori contemporaneamente.
Ogni impiegato che afferisce ad un laboratorio prende parte automaticamente a tutti i progetti a cui quel laboratorio sta lavorando.

Il manager scientifico di un laboratorio decide a quali progetti partecipare e abbandonare, e richiede attrezzatura.

Le richieste vengono memorizzate nell'entità *equipment_request*. Queste vengono valutate dal manager e dal referente scientifico del progetto a cui sono state fatte, i quali decidono, in base ai fondi del progetto, se soddisfare le richieste, con il vincolo che il costo totale delle attrezzature non può superare il 50% dei fondi del progetto.

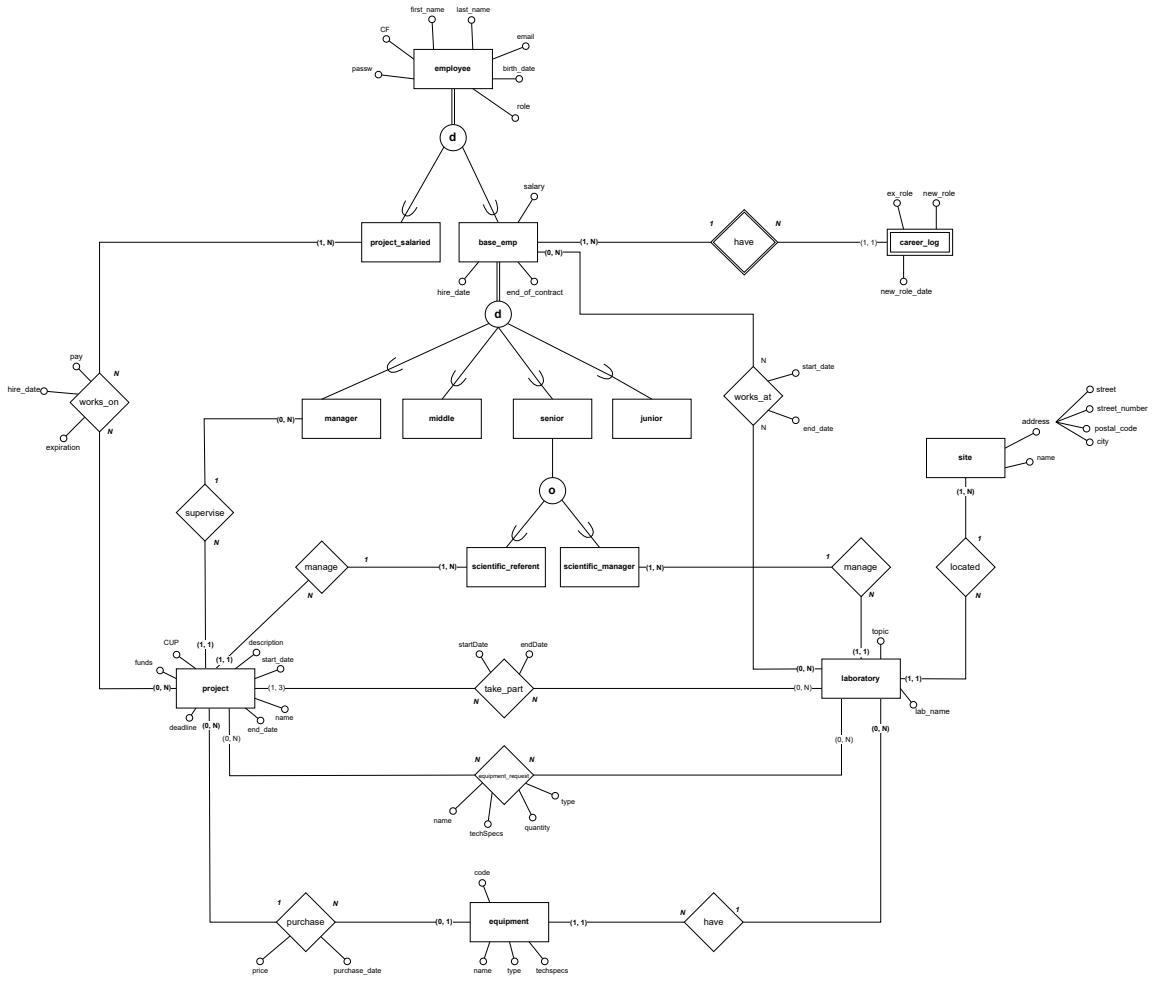
Quando viene acquistata dell'attrezzatura, si salva l'acquisto nell'entità *purchase* che tiene traccia della data e del prezzo d'acquisto.

Ogni attrezzatura di ogni laboratorio viene memorizzata nell'entità *equipment*.
Un laboratorio può possedere già dell'attrezzatura (acquistata precedentemente da altri progetti) e riceverne altra tramite richieste a progetti. Di conseguenza l'attrezzatura acquistata in un laboratorio, rimane anche dopo la fine del progetto.

Con il restante 50% dei fondi, il manager e il referente scientifico di un progetto possono assumere del personale che viene pagato per lavorare esclusivamente per quel progetto, e non viene considerato come impiegato regolare.

Lo stesso impiegato può essere assunto più volte (anche per lo stesso progetto ma in date differenti), per questo motivo ognuno di questi impiegati non viene rimosso una volta terminato il contratto. La relazione *works_on* infatti, tiene traccia di tutte le volte in cui un impiegato è stato assunto per lavorare ai progetti.

## 1.2 Schema concettuale



## 1.3 Dizionario delle entità

| Entità | Descrizione | Attributi |
|---|---|---|
| employee | Generico impiegato dell'azienda | **cf**: Codice fiscale, informazione utile all'azienda<br>**first_name**<br>**last_name**<br>**birth_date**<br>**role**: Professione (sviluppatore, designer, …)<br>**email**: Email necessaria per l'accesso al gestionale<br>**passw**: Password necessaria per l'accesso al gestionale |
| base_emp | Specializzazione di *employee*. Assunto a tempo indeterminato. | **salary**: Stipendio<br>**hire_date**: Data assunzione |
| career_log | Entità che memorizza gli scatti di carriera degli impiegati *base_emp* | **ex_role**: Ruolo precedente (stringa vuota quando è assunto)<br>**new_role**: Ruolo successivo (stringa vuota quando è licenziato)<br>**new_role_date**: Data scatto di carriera |
| project_salaried | Specializzazione di *employee*. Assunto per lavorare esclusivamente ad un progetto. | |
| junior | Specializzazione di *base_emp*. Impiegato che lavora da meno di 3 anni. | |
| middle | Specializzazione di *base_emp*. Impiegato che lavora dai 4 ai 7 anni. | |
| senior | Specializzazione di *base_emp*. Impiegato che lavora da più di 7 anni. | |
| manager | Specializzazione di *base_emp*. Impiegato che ha ricevuto una promozione in base al merito. | |
| scientific_referent | Specializzazione di *senior*. Impiegato a cui è stata affidata la gestione di un progetto. | |
| scientific_manager | Specializzazione di *senior*. Impiegato a cui è stata affidata la gestione di un laborfatorio. | |

| | | |
|---|---|---|
| project | Progetto a cui prendono parte i laboratori e i relativi impiegati. | **CUP**: Codice Univoco Progett<br>**name**: Nome del progetto<br>**description**: Descrizione del progetto<br>**start_date**: Data di inizio del progetto<br>**end_date**: Data di effettiva fine del progetto<br>**deadline**: Data prevista per la fine del progetto<br>**funds**: Somma di denaro disponibile per l'acquisto di attrezzatura e personale |
| laboratory | Luogo all'interno dell'azienda nel quale gli impiegati lavorano ai progetti. | **topic**: Campo di studi del laboratorio<br>**lab_name**: Nome del laboratorio |
| site | Sede dell'azienda in cui possono essere presente i laboratori. | **name**: Nome della sede<br>**address**: Indirizzo<br>• **street**: Via<br>• **street_number**: Numero civico<br>• **postal_code**: CAP<br>• **city**: Città |
| equipment | Attrezzatura presente all'interno di un laboratorio | **code**: Codice del prodotto<br>**name**: Nome del prodotto<br>**type**: Tipo di prodotto (computer, microscopio, …)<br>**techspecs**: Specifiche tecniche del prodotto |

## 1.4 Dizionario delle associazioni

| Associazioni | Descrizione | Attributi |
|---|---|---|
| works_on | Associazione tra *project_salaried* e *project* **molti-a-molti** | **pay**: Paga stabilita per lavorare a quel progetto<br>**hire_date**: Data di assunzione<br>**expiration**: Data di fine contratto |
| works_at | Associazione tra *base_emp* e *laboratory* **molti-a-molti**. | **start_date**: Data di assegnazione di un impiegato ad un laboratorio<br>**end_date**: Data di fine lavoro |
| supervise | Associazione tra *manager* e *project* **uno-a-molti**. | |
| manage | Associazione tra *scientific_referent* e *project* **uno-a-molti**. | |
| manage | Associazione tra *scientific_manager* e *laboratory* **uno-a-molti**. | |
| take_part | Associazione tra *project* e *laboratory* **molti-a-molti**. | **start_date**: Data in cui un laboratorio inizia a lavorare ad un progetto<br>**end_date**: Data di fine lavoro |
| purchase | Associazione tra *project* e *equipment* **uno-a-molti**. | **purchase_date**: Data di acquisto<br>**price**: Prezzo dell'attrezzatura |
| equipment_request | Associazione tra *project* e *laboratory* **molti-a-molti**. | |
| have | Associazione tra *equipment* e *laboratory* **uno-a-molti**. | |
| have | Associazione tra *base_emp* e *career_log* **uno-a-molti**. | |
| located | Associazione tra *site* e *laboratory* **uno-a-molti**. | |

# 2 Ristrutturazione del modello concettuale

## 2.1 Analisi delle ridondanze

È presente una ridondanza nel campo *hire_date* di *base_emp* in quanto è possibile calcolarla dall'entità associata *career_log*

## 2.2 Eliminazione degli attributi multivalore

Non sono presenti attributi multivalore

## 2.3 Eliminazione degli attributi composti

L'attributo *address* di *site* è un attributo composto.

Siccome l'indirizzo è unico per ogni sede, si è deciso di accorpare gli attributi di *address* in *site*.

L'attributo *tech_specs* di *equipment* potrebbe essere scomposto in più attributi ma in questo caso non è necessario ai fini della traccia, quindi è più comodo memorizzarlo come un unico campo stringa.

## 2.4 Analisi delle generalizzazioni

Procediamo all'eliminazione delle generalizzazioni partendo dal basso:

1. Accorpamento di *scientific_referent* e *scientific_manager* in *senior* in quanto non avendo attributi non è possibile inserire valori nulli che occupano spazio inutile.
   Fatta tale scelta bisogna cambiare la cardinalità da $(1, n)$ a $(0, n)$ in entrambe le associazioni, poiché un *senior* può anche non essere nessuno dei due o uno soltanto (overlapping parziale)

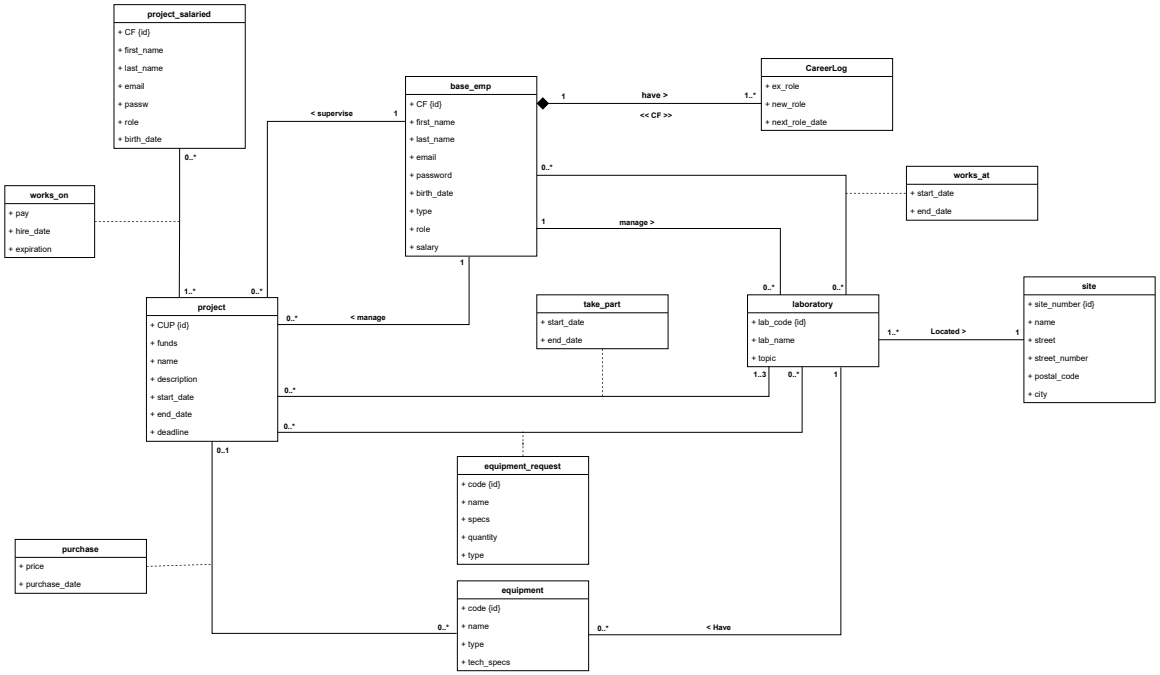
2. Accorpamento di *junior*, *middle*, *senior*, *manager* all'interno di *base_emp* aggiungendo un attributo che ne specifica il tipo.
   Notare che questa scelta aggiunge una **ridondanza** poiché il tipo è reperibile effettuando un'interrogazione in *career_log*. Nonostante ciò è più efficiente e immediato eseguire determinate operazioni avendo quest'attributo disponibile direttamente in *base_emp*

3. Accorpamento di *employee* all'interno di *project_salaried* e *base_emp* in quanto la generalizzazione è totale.

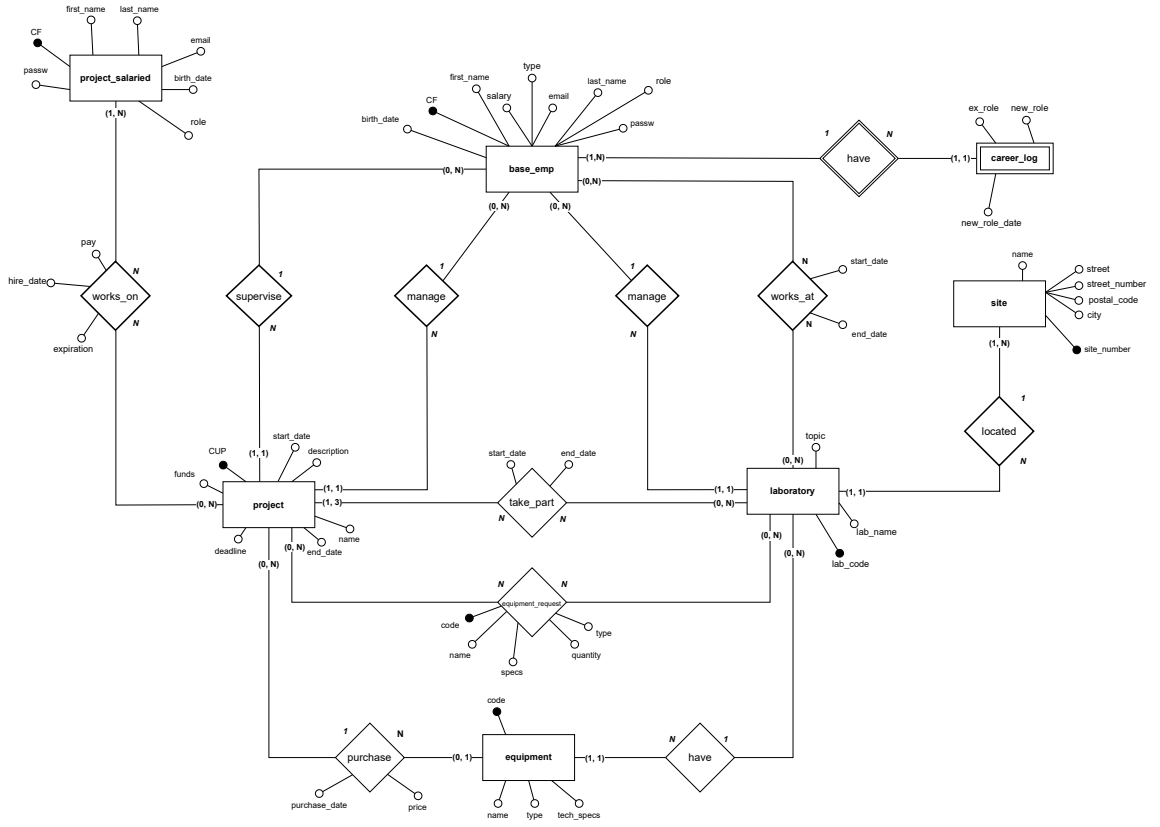
## 2.5 Identificazioni chiavi primarie

- In *base_emp* abbiamo l'attributo *CF* (Codice Fiscale), una stringa di 16 caratteri
- In *project_salaried* abbiamo l'attributo *CF* (Codice Fiscale), una stringa di 16 caratteri
- *project* è identificato dal *CUP* (Codice Univoco Progetto)
- *equipment* è identificato da *code* che è un codice generato randomicamente
- *equipment_request* è identificato da *code* che è un codice generato randomicamente
- *site* è identificato da un nuovo attributo *site_number* che si autoincrementa
- *laboratory* è identificato da un nuovo attributo *lab_code* che si autoincrementa

## 2.6 Schema ristrutturato UML




## 2.7 Schema ristrutturato ER



# 3 Traduzione al modello logico

## 3.1 Mapping associazioni

### 3.1.1 Associazioni 1-N

- *Site* - *Located* - *Laboratory*: inserimento chiave di *Site* in *Laboratory* come chiave esterna
- *base_emp* - *Supervise* - *Project*: inserimento chiave di *base_emp* all'interno di *Project* come chiave esterna
- *base_emp* - *Manage* - *Project*: inserimento chiave di *base_emp* all'interno di *Project* come chiave esterna
- *base_emp* - *Manage* - *Laboratory*: inserimento di chiave di *base_emp* all'interno di *Laboratory* come chiave esterna
- *Laboratory* - *Have* - *Equipment*: inserimento di chave di *Laboratory* in *Equipment* come chiave esterna
- *Project* - *Purchase* - *Equipment*: Equipment ha una partecipazione parziale, si procede come N-N
- *base_emp* - *Have* - *career_log*: relazione identificante, inserimento di chiave primaria di *base_emp* in *career_log* come chiave esterna

### 3.1.2 Associazioni N-N

Per ognuna di queste relazioni, si inseriscono le chiavi delle due associazioni come chiavi esterne.

| **Associazione** | **Relazione** | **Associazione** |
|---|---|---|
| project_salaried | works_on | project |
| project | take_part | laboratory |
| project | equipment_request | laboratory |
| base_emp | works_at | laboratory |

## 3.2 Modello logico

Gli attributi sottolineati sono chiavi primarie,
gli attributi con un asterisco * alla fine sono chiavi esterne

***base_emp*** (CF, first_name, last_name, email, passw, birth_date, type, role, salary)

***career_log*** (lab_code, lab_name, topic, cf_scientific_manager*, site_number*)
cf_scientific_manager ↦ base_emp.CF
site_number ↦ site.site_number

***laboratory*** (lab_code, lab_name, topic, cf_scientific_manager*, site_number*)
cf_scientific_manager ↦ base_emp.CF
site_number ↦ site.site_number

***take_part*** (start_date, end_date, CUP*, lab_code*)
CUP ↦ project.CUP
lab_code ↦ laboratory.lab_code

***project_salaried*** (CF, first_name, last_name, email, passw, birth_date, role)

***works_on*** (pay, hire_date, expiration, CF*, CUP*)
CF ↦ project_salaried.CF
CUP ↦ project.CUP

***site*** (site_number, name, street, street_number, postal_code, city)

***equipment*** (code, name, type, tech_specs, lab_code*)
lab_code ↦ laboratory.lab_code

***purchase*** (purchase_date, price, CUP*, equipment_code*)
CUP ↦ project.CUP
equipment_code ↦ equipment.code

***works_at*** (start_date, end_date, cf_base_emp*, lab_code*)
cf_base_emp↦ base_emp.CF
lab_code ↦ laboratory.lab_code

***equipment_request*** (code, type, name, specs, quantity, CUP*, lab_code*)
CUP ↦ project.CUP
lab_code ↦ laboratory.lab_code

***project*** (CUP, funds, name, description, start_date, end_date,
deadline, cf_manager*,cf_scientific_referent*)
cf_manager ↦ base_emp.CF
cf_scientific_referent ↦ base_emp.CF

# 4 Progettazione fisica

## 4.1 Creazione del database

```
CREATE DATABASE company WITH OWNER postgres;
```

## 4.2 Creazione dello schema

```
CREATE SCHEMA projects_schema;
```

## 4.3 Attivazione estensioni

Le estensioni possono essere attivate solo da un superuser (*postgres* in questo caso); *project_admin* non può essere superuser altrimenti avrebbe gli stessi privilegi di *postgres*.

Le estensioni vengono create sullo schema *projects_schema* così che l'utente *project_admin* ne abbia l'accesso.

Estensione utilizzata per la cifratura delle password, usata per l'autenticazione lato client.

```
CREATE EXTENSION IF NOT EXISTS "pgcrypto" SCHEMA projects_schema;
```

Estensione utilizzata per generare codici univoci causali.

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp" SCHEMA projects_schema;
```

## 4.4 Gestione ruoli e permessi

## 4.5 Creazione domini

```
CREATE DOMAIN cf_type AS VARCHAR(16)
CONSTRAINT cf_type_length CHECK (length(VALUE) = 16);
```

```
CREATE DOMAIN cup_type AS VARCHAR(15)
CONSTRAINT check_cup_type CHECK ( length(VALUE) = 15);
```

```
CREATE DOMAIN emp_type AS VARCHAR(10)
CONSTRAINT emp_type_check CHECK
    (VALUE IN ('', 'junior', 'middle', 'senior', 'manager')
        AND VALUE IS NOT NULL);
```

```
CREATE DOMAIN equipment_name_type
AS VARCHAR(30);
```

```sql
CREATE DOMAIN name_type AS VARCHAR(30);
```

```sql
CREATE DOMAIN password_type AS VARCHAR(100);
```

```sql
CREATE DOMAIN salary_type AS DECIMAL(8,2);
```

## 4.6 Creazione tabelle

### Tabella base_emp

```sql
CREATE TABLE base_emp(
    CF          cf_type,
    first_name  name_type     NOT NULL,
    last_name   name_type     NOT NULL,
    email       VARCHAR(100)  NOT NULL,
    passw       password_type NOT NULL,
    birth_date  DATE          NOT NULL,
    "type"      emp_type      NOT NULL,
    "role"      VARCHAR(30)   NOT NULL,
    salary      salary_type   NOT NULL,

    CONSTRAINT emp_email_unique UNIQUE (email),
    CONSTRAINT base_emp_pk PRIMARY KEY (CF)
);
```

### Tabella career_log

```sql
CREATE TABLE career_log(
    ex_role         emp_type,
    new_role        emp_type,
    new_role_date   TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    CF              cf_type,

    -- foreign key constraint
    CONSTRAINT emp_career_fk FOREIGN KEY (CF) REFERENCES base_emp(CF)
        ON DELETE CASCADE
        ON UPDATE CASCADE,

    CONSTRAINT check_new_grade CHECK (
        (ex_role, new_role) IN (
            ('', 'junior'), -- assunto
            ('junior', 'middle'),
            ('middle', 'senior')
        ) OR

        -- licenziato
        ex_role <> '' AND new_role = ''
        OR

        -- avanzamento a manger
        ex_role <> '' AND ex_role <> 'manager' AND new_role = 'manager'
        OR

        -- declassato
        --
        -- un trigger verificherà che il new_role sia quello del log precedente
        -- se da middle è diventato manager e poi viene declassato,
```

```
        --    sarà declassato a middle
        ex_role = 'manager' AND new_role <> 'manager'
    )
```

### Tabella laboratory

```
CREATE TABLE laboratory(
    lab_code    SERIAL,
    lab_name    VARCHAR(200)    NOT NULL,
    topic       VARCHAR(1000)   NOT NULL,

    -- foreign keys
    site_number             SERIAL  NOT NULL,
    cf_scientific_manager   cf_type NOT NULL,

    CONSTRAINT lab_pk                   PRIMARY KEY (lab_code),

    CONSTRAINT site_number_fk           FOREIGN KEY (site_number)
                                        REFERENCES site(site_number),

    CONSTRAINT cf_scientific_manager_fk FOREIGN KEY (cf_scientific_manager)
                                        REFERENCES base_emp(cf)
);
```

### Tabella take_part

```
CREATE TABLE take_part(
    start_date  DATE    NOT NULL,
    end_date    DATE,

    -- foreign keys
    CUP         cup_type,
    lab_code    SERIAL,

    CONSTRAINT date_integrity CHECK (end_date IS NULL OR end_date > start_date),

    CONSTRAINT CUP_fk FOREIGN KEY (CUP) REFERENCES project(CUP)
    ON UPDATE CASCADE
    ON DELETE CASCADE,

    CONSTRAINT lab_code_fk FOREIGN KEY (lab_code) REFERENCES laboratory(lab_code)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

### Tabella project_salaried

```
CREATE TABLE project_salaried(
    CF          cf_type,
    first_name  name_type       NOT NULL,
    last_name   name_type       NOT NULL,
    email       VARCHAR(100)    NOT NULL,
    passw       password_type   NOT NULL,
    "role"      VARCHAR(30)     NOT NULL,
    birth_date  DATE            NOT NULL,

    CONSTRAINT project_salaried_email_unique UNIQUE (email),
    CONSTRAINT project_salaried_pk PRIMARY KEY (CF)
);
```

### Tabella works_on

```
CREATE TABLE works_on (
    pay         salary_type NOT NULL,
    hire_date   DATE        NOT NULL,
    expiration  DATE        NOT NULL,

    -- foreign keys
    CF          cf_type,
    CUP         cup_type,

    CONSTRAINT check_expiration_date CHECK ( expiration > hire_date ),

    CONSTRAINT fk_cf FOREIGN KEY (CF) REFERENCES project_salaried(CF)
    ON UPDATE CASCADE

    -- mi interessa salvare i contratti anche se
    -- un impiegato a progetto viene eliminato
    ON DELETE SET NULL,

    CONSTRAINT fk_cup FOREIGN KEY (CUP) REFERENCES project(CUP)
    ON UPDATE CASCADE
);
```

### Tabella site

```
CREATE TABLE site(
    site_number     SERIAL,
    name            VARCHAR(40) NOT NULL,
    street          VARCHAR(30) NOT NULL,
    street_number   VARCHAR(10) NOT NULL,
    postal_code     VARCHAR(10) NOT NULL,
    city            VARCHAR(20) NOT NULL,

    CONSTRAINT site_pk PRIMARY KEY (site_number)
);
```

### Tabella equipment

```
CREATE TABLE equipment(
    code        uuid                DEFAULT uuid_generate_v4(),
    name        equipment_name_type NOT NULL,
    type        VARCHAR(30)         NOT NULL,
    tech_specs  VARCHAR(100),

    -- foreign keys
    lab_code    SERIAL,

    CONSTRAINT equipment_pk PRIMARY KEY (code),

    CONSTRAINT lab_code_fk FOREIGN KEY (lab_code) REFERENCES laboratory(lab_code)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);
```

### Tabella purchase

```
CREATE TABLE purchase(
    price           DECIMAL(10,2)   NOT NULL,
```

```
    purchase_date   DATE            NOT NULL,

    -- foreign keys
    CUP             cup_type,
    equipment_code  uuid            DEFAULT uuid_generate_v4(),

    CONSTRAINT cup_type_fk FOREIGN KEY (CUP) REFERENCES project(CUP)
        ON UPDATE CASCADE
        ON DELETE SET NULL,

    CONSTRAINT equipment_code_fk    FOREIGN KEY (equipment_code)
                                    REFERENCES equipment(code)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT check_price CHECK ( price > 0)
);
```

### Tabella works_at

```
CREATE TABLE works_at(
    start_date  DATE    NOT NULL,
    end_date    DATE,

    -- foreign keys
    cf_base_emp cf_type,
    lab_code    SERIAL,

    CONSTRAINT date_integrity CHECK (end_date IS NULL OR end_date > start_date),

    CONSTRAINT cf_base_emp_fk FOREIGN KEY (cf_base_emp) REFERENCES base_emp(cf)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT lab_code_fk    FOREIGN KEY (lab_code) REFERENCES laboratory(lab_code)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

### Tabella equipment_request

```
CREATE TABLE equipment_request(
    code     uuid                   DEFAULT uuid_generate_v4(),

    name     equipment_name_type    NOT NULL,
    specs    VARCHAR(100)           NOT NULL,
    type     VARCHAR(30)            NOT NULL,

    quantity INTEGER                NOT NULL   DEFAULT 1,

    -- foreign keys
    CUP      cup_type,
    lab_code SERIAL,

    CONSTRAINT equipment_request_pk PRIMARY KEY (code),

    CONSTRAINT CUP_fk       FOREIGN KEY (CUP)      REFERENCES project(CUP)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT lab_code_fk  FOREIGN KEY (lab_code) REFERENCES laboratory(lab_code)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
```

```
    CONSTRAINT check_quantity CHECK ( quantity > 0 )
);
```

**Tabella project**

```
CREATE TABLE project(
    CUP             cup_type,
    funds           DECIMAL(10,2)   NOT NULL,
    "name"          VARCHAR(50),
    description     VARCHAR(100)    NOT NULL,
    start_date      DATE            NOT NULL DEFAULT CURRENT_DATE,
    end_date        DATE,
    deadline        DATE,

    -- foreign key
    cf_manager              cf_type NOT NULL,
    cf_scientific_referent  cf_type NOT NULL,

    CONSTRAINT project_pk PRIMARY KEY (CUP),

    CONSTRAINT check_deadline CHECK ( deadline IS NULL OR deadline > start_date ),
    CONSTRAINT check_end_date CHECK ( end_date IS NULL OR end_date > start_date ),

    CONSTRAINT fk_cf_manager FOREIGN KEY (CF_manager) REFERENCES base_emp(CF)
        ON UPDATE CASCADE,

    CONSTRAINT fk_cf_scientific_referent    FOREIGN KEY (CF_scientific_referent)
                                            REFERENCES base_emp(CF)
        ON UPDATE CASCADE
);
```

## 4.7 Trigger e trigger function

## 4.8 Procedure e funzioni

## 4.9 Dizionario dei vincoli

### 4.9.1 Vincoli intra-relazionali

| **base_emp** | |
|---|---|
| **Vincolo** | **Descrizione** |
| emp_email_unique | L'email è univoca |
| base_emp_pk | Vincolo di chiave primaria |

| **laboratory** | |
|---|---|
| **Vincolo** | **Descrizione** |
| lab_pk | Vincolo di chiave primaria |
| site_number_fk | Vincolo di chiave esterna |
| cf_scientific_manager_fk | Vincolo di chiave esterna |

| career_log | |
|---|---|
| **Vincolo** | **Descrizione** |
| emp_career_fk | Vincolo di chiave esterna.<br>Se viene eliminato l'impiegato, verrà eliminata anche la tupla in *career_log* |
| check_new_grade | Controlla che gli scatti di carriera siano coerenti.<br>Ad esempio non è possibile passare da *junior* a *senior* senza essere diventati prima *middle*.<br>La stringa vuota come primo parametro significa che l'impiegato è stato assunto;<br>come secondo invece, significa che è stato licenziato.<br>**Casi possibili:**<br>• (' ', 'junior')<br>• ('junior', 'middle')<br>• ('middle', 'senior')<br>• ('junior', 'manager')<br>• ('middle', 'manager')<br>• ('senior', 'manager')<br>• ('manager', 'junior')<br>• ('manager', 'middle')<br>• ('manager', 'senior')<br>• ('junior', ' ')<br>• ('middle', ' ')<br>• ('senior', ' ')<br>• ('manager', ' ') |

| equipment_request | |
|---|---|
| **Vincolo** | **Descrizione** |
| equipment_request_pk | Vincolo di chiave primaria |
| CUP_fk | Vincolo di chiave esterna.<br>Se viene eliminato il progetto (o modificato il cup), verrà eliminata (o aggiornata) anche la tupla in *equipment_request* |
| lab_code_fk | Vincolo di chiave esterna.<br>Se viene eliminato il laboratorio (o modificato il cup), verrà eliminata (o aggiornata) anche la tupla in *equipment_request* |
| check_quantity | Controlla che la quantità sia maggiore di 0 |

| **equipment** | |
|---|---|
| **Vincolo** | **Descrizione** |
| equipment_pk | Vincolo di chiave primaria |
| lab_code_fk | Vincolo di chiave esterna.<br>Se viene eliminato il laboratorio, la chiave esterna verrà posta a NULL.<br>Se viene aggiornata la chiave di *laboratory*, verrà aggiornata anche la chiave esterna. |

| **project_salaried** | |
|---|---|
| **Vincolo** | **Descrizione** |
| project_salaried_pk | Vincolo di chiave primaria |
| emp_email_unique | L'email è univoca |

| **project** | |
|---|---|
| **Vincolo** | **Descrizione** |
| project_pk | Vincolo di chiave primaria |
| check_deadline | Verifica che la deadline sia dopo la data di inizio (nel caso in cui ci fosse). |
| check_end_date | Verifica che la data di fine progetto sia dopo la data di inizio. |
| fk_cf_manager | Vincolo di chiave esterna |
| fk_cf_scientific_referent | Vincolo di chiave esterna |

| **purchase** | |
|---|---|
| **Vincolo** | **Descrizione** |
| cup_type_fk | Vincolo di chiave esterna.<br>Se viene eliminato il progetto, la chiave esterna viene posta a NULL, questo perché si vuole tenere traccia della data d'acquisto e del prezzo dell'attrezzatura. |
| equipment_code_fk | Vincolo di chiave esterna.<br>Se viene eliminato l'attrezzatura, verrà eliminata anche la tupla in purchase, dato che diventa inutile tenere traccia dell'acquisto di un prodotto che non si conosce. |
| check_price | Controlla che il prezzo sia maggiore di 0 |

| **site** | |
|---|---|
| **Vincolo** | **Descrizione** |
| site_pk | Vincolo di chiave primaria |

| **take_part** | |
|---|---|
| **Vincolo** | **Descrizione** |
| date_integrity | Verifica che la data in cui il laboratorio termina di lavorare al progetto, sia dopo la data di inizio. |
| CUP_fk | Vincolo di chiave esterna.<br>Se viene eliminato il progetto (o modificato il cup), verrà eliminata (o aggiornata) anche la tupla in *take_part* |
| lab_code_fk | Vincolo di chiave esterna.<br>Se viene eliminato il laboratorio (o modificato il lab_code), verrà eliminata (o aggiornata) anche la tupla in *take_part* |

| **works_at** | |
|---|---|
| **Vincolo** | **Descrizione** |
| date_integrity | Verifica che la data in cui l'impiegato termina di lavorare al laboratorio, sia dopo la data di inizio. |
| cf_base_emp_fk | Vincolo di chiave esterna.<br>Se viene eliminato l'impiegato (o modificato il cf), verrà eliminata (o aggiornata) anche la tupla in *works_at* |
| lab_code_fk | Vincolo di chiave esterna.<br>Se viene eliminato il laboratorio (o modificato il lab_code), verrà eliminata (o aggiornata) anche la tupla in *works_at* |

| **works_on** | |
|---|---|
| **Vincolo** | **Descrizione** |
| check_expiration_date | Verifica che la data di fine contratto sia dopo la data di assunzione. |
| fk_cf | Vincolo di chiave esterna.<br>Se viene eliminato l'impiegato, verrà posta la chiave esterna a NULL.<br>Questo perché si vuole salvare i contratti anche se un impiegato a progetto viene eliminato. |
| fk_cup | Vincolo di chiave esterna. |

| **Domini** | |
|---|---|
| **Vincolo** | **Descrizione** |
| emp_type_check | Il valore inserito dev'essere uno fra questi<br>(' ', 'junior', 'middle', 'senior', 'manager')<br>e non deve essere NULL |
| check_cup_type | Il valore inserito dev'essere lungo 15 caratteri |
| cf_type_length | Il valore inserito dev'essere lungo 16 caratteri |

#### 4.9.2 Vincoli inter-relazionai