Documentazione progetto basi di dati

Antonio Petrillo

Contents

Traccia originale

Scenario: l'organizzazione di un cantiere edile Attori:

- Amministratore del sistema: che ha possibilità di modificare tutto il database
- Capocantiere: che gestisce il cantiere di cui è responsabile

Bisogna andare a gestire l'organizzazione di un certo numero di cantieri. L'apertura di un cantiere è fatta dall'amministratore del sistema che poi ne demanda la gestione al Capocantiere. Il capocantiere dovrà tenere traccia di:

- Gli operai che lavorano nel cantiere, definendo per ognuno un ruolo all'interno del cantiere
- Aree del cantiere: ad esempio un cantiere può essere diviso in 3 aree, una zona da adibire

a verde pubblico, una da adibire a zona residenziale, un'altra da adibire a servizi pubblici.

- Ogni area del cantiere ha un operaio che ne è responsabile

Si definisca anche un ruolo "operatore", il quale ha il compito di piazzare dei sensori all'interno del cantiere. Questi sensori devono monitorare il livello di rumore in una particolare area del cantiere, e eventuali fughe di gas all'interno dell'area. Ogni area possiede una sola coppia di sensori. Il capocantiere deve essere in grado di leggere e filtrare i dati dei sensori. I dati dei sensori per ogni area e per ogni cantiere possono essere inseriti dal capocantiere o dall'amministratore. Si definisca una soglia al di sopra della quale viene lanciato un allarme (rumore troppo alto e quantità di gas molto elevata). Il capocantiere e l'amministratore possono modificare questa soglia.

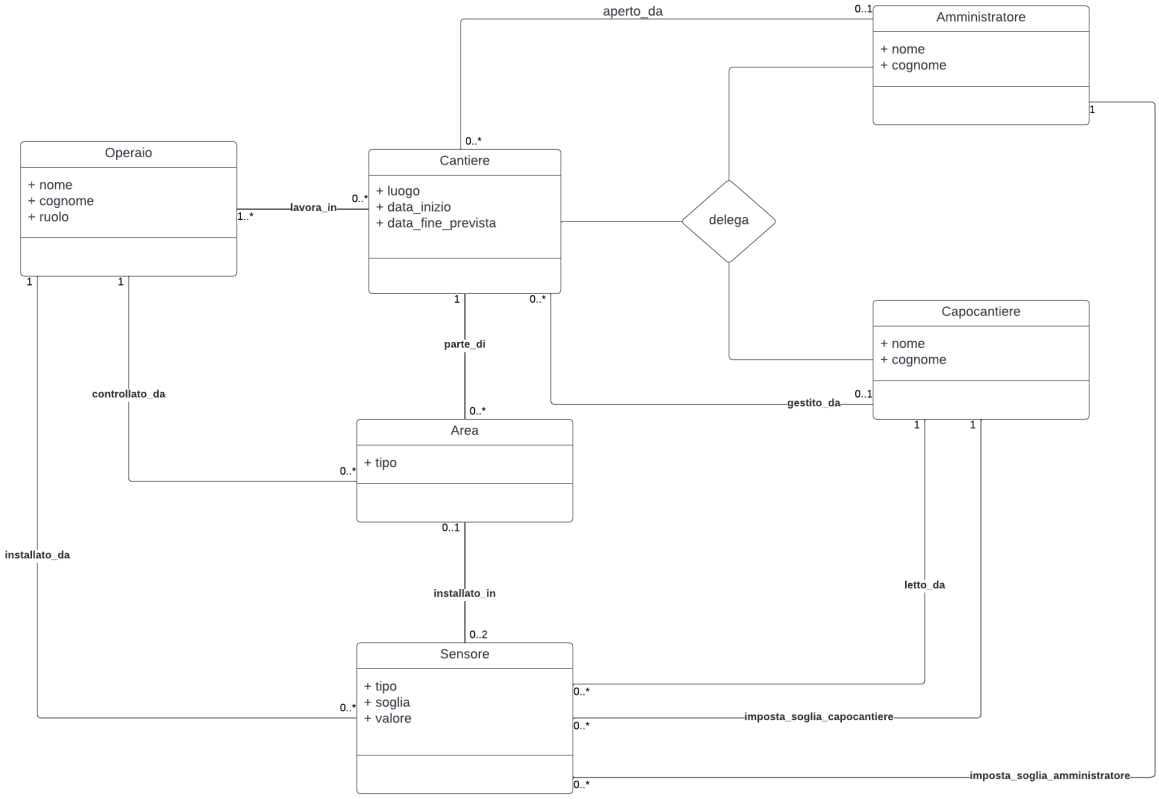
Descrizione

Progettazione di una base dati atta a contenere informazioni di cantieri edili.
Un cantiere é caratterizzato nel seguente modo:

- **Amministratore**, avvia il progetto di un cantiere. Esso puó aprire un nuovo cantiere e delegarne il controllo ad un *capocantiere*, inoltre puó modificare la *soglia* al di sopra delle quale i *sensori* lanciano un allarme
- **Capocantiere**, gestisce un particolare cantiere. Il suo compito principale é quello di leggere i *sensori* del cantiere a cui é assegnato, inoltre anche esso puó modificare la *soglia* di allarme dei *sensori*.
- **Cantiere**. Ogni cantiere é suddiviso in piú *aree*.
- **Area**. Area di un determinato **cantiere**. Ogni area possiede due **sensori**.
- **Sensore**. Sensore installato in una particolare **area** di un **cantiere**. Ogni sensore viene installato da un **operatore**. Ogni sensore possiede una soglia al di sopra della quale genera un allarme. Ve ne sono di due tipi:
  - Monitorarizzazione del **rumore**
  - Monitorarizzazione del **gas**
- **Operaio**. Impiegato del cantiere.

Progettazione Concettuale

Diagramma concettuale



Ristrutturazione

Rimozione ridondanze

Viene esplicitamente richiesto che un **Sensore** abbia una soglia impostabile da parte dell'amministratore e del capocantiere del progetto, questa informazione é facilmente recuperabile controllando in quale cantiere é installato, e di conseguenza da chi é stato aperto e da chi é stato gestito, le stesse osservazioni si possono fare per l'associazione *leggibile da*. Questa informazione pertanto verrá rimossa.

Attributi multipli

Nella classe **Cantiere** vi é l'attributo *luogo* che ha vari campi essendo composto da, indirizzo, CAP, numero civico. Per risolvere questo problema verrá introdotta una nuova classe **Luogo** che conterrá i campi sopra citati.

Entitá deboli

Nella classe **Sensore** vi é l'attributo *valore* che contiene appunto la lettura del dispositivo. Questo campo é sufficiente a rappresentare l'informazione richiesta ma con un cavillo, é possibile contenere solo l'ultima lettura effettua, perdendo cosí quelli letti in precedenza. Per risolvere questo problema si é optato per inserire una nuova classe per rappresentare i valori letti, a quest'ultimi viene associato il timestamp di scrittura cosí da poterli distinguere.

Generalizzazioni

Anche se non evidente dal diagramma, esistono due tipi di sensori distinti, sensori di rumore e sensori per il gas. Hanno entrambi gli stessi attributi quindi si é preferito optare per una classe generale che possa rappresentare entrambe, le due tipologie vengono poi discriminate tramite una enumerazione.

Enumerazioni

- Ruolo L'attributo *ruolo* della classe **Operaio** puó avere solo alcuni valori predefinti, quindi si é optato per l'introduzione di una enumerazione i valori di quest'ultima sono i seguenti:
  - SEMPLICE
  - IDRAULICO
  - MACCHINISTA
  - ELETTRICISTA
  - OPERATORE
- Zona L'attributo *tipo* della classe **Area** viene modellato come una enumerazione con i seguenti valori:
  - SERVIZIPUBBLICI
  - ZONAVERDE

  - ZONARESIDENZIALE
  - ZONARISTORAZIONE

- Tipo Serve ad identificare il tipo di sensore installato in una particolare area:
  - GAS
  - RUMORE

Ricerca identificativi

Nessuna delle classi presenti possiede degli attributi tali da poter identificare le singole istanze, é necessario introdurre per ognuno una chiave tecnica. L'unica eccezione é la classe **Luogo** che peró necessita di una chiave primaria composta da tre campi, per questo motivo si é deciso di inserire una chiave tecnica anche qui.

Dizionario delle classi

- Cantiere
  - Descrizione Descrittore di ogni cantiere presente nella base dati.
  - Attributi
    - id (Integer): Chiave tecnica, univoca per ogni cantiere.
    - data_inizio (Date): Data in cui il cantiere é stato aperto.
    - data_fine_prevista (Date): Data in cui si presuma i lavori debbano terminare.

- Luogo
  - Descrizione Descrittore della posizione geografica di un cantiere.
  - Attributi
    - indirizzo (String): Indirizzo del cantiere.
    - $\text{numero}_{\text{civico}}$ (Integer): Numero civico dell'indirizzo.
    - CAP (String): CAP del luogo in cui é situato il cantiere.
    - cittá (String): cittá in cui a cui fa riferimento l' indirizzo

- Area
  - Descrizione Descrittore delle aree di cui é composto un cantiere.

  - Attributi
    * id (Integer): Chiave tecnica, univoca per ogni area.
    * tipo (Zona): Descrive che tipo di area é in costruzione.

- Amministratore
  - Descrizione Descrittore di un amministratore.
  - Attributi
    * id (Integer): chiave tecnica, univoca per ogni amministratore.
    * nome (String): Nome dell'amministratore.
    * cognome (String): Cognome dell'amministratore.

- Capocantiere
  - Descrizione Descrittore di un capocantiere.
  - Attributi
    * id (Integer): Chiave tecnica, univoca per ogni capocantiere.
    * nome (String): Nome del capocantiere.
    * cognome (String): Cognome del capocantiere.

- Operaio
  - Descrizione Descrittore di un operaio che viene impiegato in un cantiere.
  - Attributi
    * id (Integer): Chiave tecnica, univoca per ogni operaio.
    * nome (String): Nome dell'operaio.
    * cognome (String): Cognome dell'operaio.
    * ruolo (Ruolo): Lavoro in cui é specializzato l'operaio.

- Sensore
  - Descrizione Descrittore di un sensore installato in un'area di un cantiere.
  - Attributi
    * id (Integer): Chiave tecnica, univoca per ogni sensore.
    * dati (Number): Dati che verranno letti dal sensore.
    * soglia (Number): Soglia oltre il quale il sensore lancerá un allarme.

        * tipo (Tipo): Tipo di sensore installato.

- Valore
  - Descrizione Entitá debole associata alle letture di un sensore.
  - Attributi
    * $\text{data}_{\text{scrittura}}$ (Date): data di scrittura del valore
    * valore (Number): magnitudine del valore

Dizionario delle associazioni

- apre
  - Descrizione Indica quale amministratore ha aperto un determinato cantiere.
  - Classi coinvolte
    * Amministratore [0..1]
    * Cantiere [0..*]

- delega
  - Descrizione Indica quale **Cantiere** viene assegnato ad un **Capocantiere** da parte di un **Amministratore**.
  - Classi coinvolte
    * Amministratore [0..*]
    * Capocantiere [0..*]
    * Cantiere [0..*]

- installato_in
  - Descrizione Indica quale **Sensore** é installato in un determinato **Cantiere**.
  - Classi coinvolte
    * Sensore [0..2]
    * Area [0..1]

- composto_da
  - Descrizione Indica da quale **Aree** é composto un **Cantiere**.
  - Classi coinvolte

        * Area [0..*]
        * Cantiere [1]

- lavora_in
    - Descrizione Indica quale **Operaio** lavora in un determinato **Cantiere**.
    - Classi coinvolte
        * Cantiere [0..*]
        * Operaio [1..*]

- responsabile
    - Descrizione Indica quale **Operaio** é responsabile di una determinata **Area**.
    - Classi coinvolte
        * Operaio [1]
        * Area [0..*]

- installa
    - Descrizione Indica quale **Operaio** ha installato un determinato **Sensore**.
    - Classi coinvolte
        * Operaio [1]
        * Sensore [0..*]

Dizionario dei vincoli

- delega_univoca
    - Tipo Interelazionale.
    - Descrizione Un **Cantiere** aperto da un **Amministratore** non puó essere delegato a piú **Capocantieri**.

- valore_non_supera_soglia
    - Tipo Interelazionale.
    - Descrizione Il *valore* associato ad un **Sensore** non puó superare la *soglia*.

- data_fine_plausibile

  - Tipo Intrarelazionale.
  - Descrizione La data contenuta in *data_fine_prevista* deve essere successiva a *data_inizio* in **Cantiere**.

- data_nuova_scrittura
  - Tipo Intrarelazionale.
  - Descrizione Quando un nuovo valore, associato ad un sensore, viene inserito nella specifica tabella, é necessario che la data di scrittura sia piú recente dell'ultima inserita.

- amministratore_corretto_delega
  - Tipo Interelazionale.
  - Descrizione Un cantiere puó essere delegato solo dall'amministratore che lo ha aperto.

- solo_operatore_installa_sensore
  - Tipo Interelazionale.
  - Descrizione Un **Sensore** puó essere installato solo da un **Operatore**.

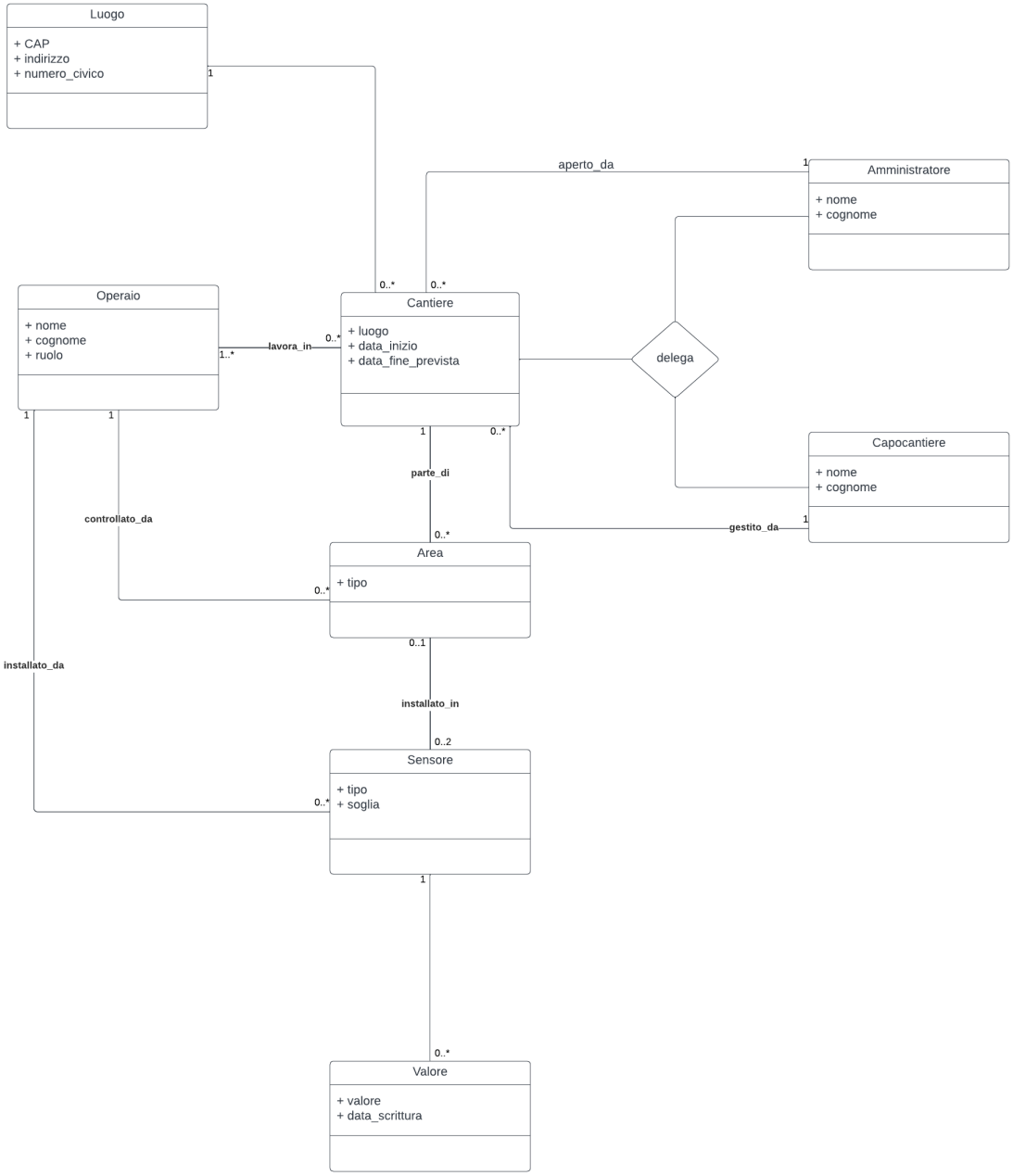
- numero_civico_naturale
  - Tipo Intrarelazionale.
  - Descrizione Il $numero_{civico}$ di un **Luogo** deve essere un numero maggiore di zero.

- max_due_sensori_differenti
  - Tipo Interrelazionale.
  - Descrizione Ad ogni area possono essere associati solo sensori di tipo differente fra loro. Questo vincolo impone anche che ad ogni **Area** siano associati al piú due sensori dato che esistono solo due tipi di questi.

- CAP_ha_5_caratteri
  - Tipo Intrarelazionale
  - Descrizione Il *CAP* di un luogo deve essere composto da cinque caratteri.

- CAP_composto_solo_da_cifre
  - Tipo Intrarelazionale
  - Descrizione Il *CAP* deve essere composto solo da cifre da **0** a **9**.
- luogo_univoco
  - Tipo Intrarelazionale
  - Descrizione Non esistono luoghi con tutti i campi uguali fatta eccezione per l'ID.
- area_univoca_per_cantiere
  - Tipo Interelazionale.
  - Descrizione Un cantiere non puó avere piú **Aree** dello stesso tipo.
- data_scrittura_dopo_inizio_lavori
  - Tipo Interelazionale.
  - Descrizione La scrittura di un *valore* di un **Sensore** nella tabella **Valore** deve essere conseguente alla data di inizio lavori.
- valore_scritto_positivo
  - Tipo Intrarelazionale
  - Descrizione Il valore scritto in un sensore deve essere positivo.

Diagramma ristrutturato



Progettazione Logica

Convenzione

Nella prossima sezione verranno indicate con una singola sottolineatura le chiavi *primarie*, mentre le chiavi *esterne* con una doppia sottolineatura.

Traduzione

- **Cantiere** (*id*, data_inizio, data_fine_prevista, *id_luogo*, *aperto_da*, *gestito_da*)
  - **aperto da** → Amministratore.*id*
  - **gestito da** → Capocantiere.*id*
  - *id_luogo* → Luogo.*id*
- **Area** (*id*, tipo, *parte_di*, *controllato_da*)
  - **parte di** → Cantiere.*id*
  - **controllato da** → Operaio.*id*
- **Sensore** (*id*, tipo, *installato_da*, *installato_in*)
  - **installato da** → Operaio.*id*
  - **installato in** → Area.*id*
- **Valore** (valore, *data_scrittura*, *valore_di*)
  - **valore di** → Sensore.*id*
- **Amministratore** (*id*, nome, cognome)
- **Capocantiere** (*id*, nome, cognome)
- **Operaio** (*id*, nome, cognome, ruolo)
- **Luogo** (*id*, CAP, indirizzo, *numero_civico*)
- **Delega** (*cantiere_id*, *amministratore_id*, *capocantiere_id*)
  - *cantiere_id* → Cantiere.*id*
  - *amministratore_id* → Amministratore.*id*
  - **capocantiere_id** → Capocantiere.*id*

- **Lavora in** (*operaio_id*, *cantiere_id*)
  - *cantiere_id* → Cantiere.*id*
  - *operaio_id* → Operaio.*id*

Progettazione Fisica

Scelta del DBMS

La base dati é stata realizzata con il DBMS Postgres. Una peculiaritá di questo DBMS é che non implementa le ASSERTION, queste sono state implementate tramite PROCEDURES e TRIGGER. In particolare per implementare una assertion vi é una procedura che si occupa di effettuare il controllo del vincolo e un trigger il cui compito é chiamare la procedura. Tutti i dettagli possono essere consultati nella sezione successiva dell'SQL.

Creazione Database

```
CREATE DATABASE cantiere
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.utf8'
    LC_CTYPE = 'en_US.utf8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;
```

Creazione dello schema

```
CREATE SCHEMA IF NOT EXISTS cantiere
    AUTHORIZATION postgres;
```

Definizione enumerazioni

- Enumerazione ruolo

```
CREATE TYPE cantiere.ruolo AS ENUM
    ('semplice', 'idraulico', 'macchinista', 'elettricista', 'operatore');
```

- Enumerazione zona

```
CREATE TYPE cantiere.zona AS ENUM
    ('servizi_pubblici', 'zona_verde', 'zona_residenziale', 'zona_ristorazione');
```

- Enumerazione tipo

```
CREATE TYPE cantiere.tipo AS ENUM
       ('gas', 'rumore');
```

Definizione tabelle

- Tabella amministratore

```
CREATE TABLE cantiere.amministratore(
       id SERIAL PRIMARY KEY,
       nome VARCHAR(50) NOT NULL,
       cognome VARCHAR(50) NOT NULL);
```

- Tabella capocantiere

```
CREATE TABLE cantiere.capocantiere(
       id SERIAL PRIMARY KEY,
       nome VARCHAR(50) NOT NULL,
       cognome VARCHAR(50) NOT NULL);
```

- Tabella luogo

```
CREATE TABLE cantiere.luogo(
       id SERIAL PRIMARY KEY,
       citta VARCHAR(50) NOT NULL,
       CAP VARCHAR(5) NOT NULL,
       indirizzo VARCHAR(50) NOT NULL,
       numero_civico INT);
```

- Tabella cantiere

```
CREATE TABLE cantiere.cantiere(
       id SERIAL PRIMARY KEY,
       data_inizio DATE NOT NULL,
       data_fine_prevista DATE NOT NULL,
       aperto_da SERIAL,
       luogo_id SERIAL,
```

```
        FOREIGN KEY (luogo_id)
                REFERENCES
                 cantiere.luogo(id)
                     ON UPDATE CASCADE,
        FOREIGN KEY (aperto_da)
                REFERENCES
                 cantiere.amministratore(id)
                     ON UPDATE CASCADE);
```

- Tabella operaio

```
CREATE TABLE cantiere.operaio(
       id SERIAL PRIMARY KEY,
       nome VARCHAR(50) NOT NULL,
       cognome VARCHAR(50) NOT NULL,
       ruolo cantiere.RUOLO);

ALTER TABLE cantiere.operaio
            ALTER COLUMN ruolo SET DEFAULT 'semplice';
```

- Tabella area

```
CREATE TABLE cantiere.area(
       id SERIAL PRIMARY KEY,
       zona cantiere.ZONA NOT NULL,
       parte_di SERIAL,
       controllato_da SERIAL,
       FOREIGN KEY(parte_di)
               REFERENCES cantiere.cantiere(id)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE,
       FOREIGN KEY(controllato_da)
               REFERENCES cantiere.operaio(id)
                          ON UPDATE CASCADE);
```

- Tabella sensore

```
CREATE TABLE cantiere.sensore(
       id SERIAL PRIMARY KEY,
       tipo cantiere.TIPO NOT NULL,
```

```
        soglia FLOAT8 NOT NULL,
        installato_da SERIAL,
        installato_in SERIAL,
        FOREIGN KEY(installato_da)
                REFERENCES cantiere.operaio(id)
                           ON UPDATE CASCADE
                           ON DELETE SET NULL,
        FOREIGN KEY(installato_in)
                REFERENCES cantiere.area(id)
                           ON UPDATE CASCADE
                           ON DELETE CASCADE);
```

- Tabella valore

```
CREATE TABLE cantiere.valore(
       data_scrittura TIMESTAMP,
       valore_di SERIAL,
       valore FLOAT8 NOT NULL,
       PRIMARY KEY(data_scrittura, valore_di),
       FOREIGN KEY(valore_di)
               REFERENCES cantiere.sensore(id)
                          ON UPDATE CASCADE
                          ON DELETE CASCADE);
```

- Tabella $\text{lavora}_{\text{in}}$

```
CREATE TABLE cantiere.lavora_in(
       cantiere_id SERIAL,
       operaio_id SERIAL,
       PRIMARY KEY(cantiere_id, operaio_id),
       FOREIGN KEY(cantiere_id)
               REFERENCES cantiere.cantiere(id)
                          ON UPDATE CASCADE
                          ON DELETE CASCADE,
       FOREIGN KEY(operaio_id)
               REFERENCES cantiere.operaio(id)
                          ON UPDATE CASCADE
                          ON DELETE CASCADE);
```

- Tabella delega

```
CREATE TABLE cantiere.delega(
       cantiere_id SERIAL,
       amministratore_id SERIAL,
       capocantiere_id SERIAL,
       PRIMARY KEY(cantiere_id, amministratore_id, capocantiere_id),
       FOREIGN KEY(cantiere_id)
              REFERENCES cantiere.cantiere(id)
                         ON DELETE CASCADE
                         ON UPDATE CASCADE,
       FOREIGN KEY(capocantiere_id)
              REFERENCES cantiere.capocantiere(id)
                         ON DELETE CASCADE
                         ON UPDATE CASCADE,
       FOREIGN KEY(amministratore_id)
              REFERENCES cantiere.amministratore(id)
                         ON DELETE CASCADE
                         ON UPDATE CASCADE);
```

Definizione vincoli

Postgres non possiede un meccanismo ad-hoc per definire dei vincoli, lo stesso comportamento puó essere ottenuto tramite una procedura ed un trigger.

- Vincolo delega univoca

```
CREATE OR REPLACE FUNCTION
cantiere.delega_univoca()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
    IF EXISTS(SELECT *
              FROM cantiere.delega AS d
              WHERE d.cantiere_id = NEW.cantiere_id
                    AND d.amministratore_id = NEW.amministratore_id)
    THEN
        RAISE EXCEPTION 'il cantiere [%] é stato gía delegato', NEW.cantiere_id;
    ELSE
        RETURN NEW;
    END IF;
```

```
END;
$$

CREATE OR REPLACE TRIGGER delega_univoca_trig
BEFORE INSERT OR UPDATE ON cantiere.delega
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.delega_univoca();
```

- Vincolo valore non supera soglia

```
CREATE OR REPLACE FUNCTION
cantiere.valore_non_supera_soglia()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    soglia FLOAT8 := 0;
BEGIN
    SELECT s.soglia INTO soglia
    FROM cantiere.sensore AS s
    WHERE s.id = NEW.valore_di;

    IF soglia < NEW.valore
    THEN
        RAISE EXCEPTION '[Allarme] il sensore % ha superato la soglia: %', NEW.val
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER valore_non_supera_soglia_trig
BEFORE INSERT OR UPDATE ON cantiere.valore
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.valore_non_supera_soglia();
```

- Vincolo data fine plausibile

```
CREATE OR REPLACE FUNCTION
cantiere.data_fine_plausibile()
```

```
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
    IF NEW.data_inizio >= NEW.data_fine_prevista
    THEN
        RAISE EXCEPTION 'la data di fine lavori é antecedente a quella di inizio';
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER data_fine_plausibile_trig
BEFORE INSERT OR UPDATE ON cantiere.cantiere
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.data_fine_plausibile();
```

- Vincolo data nuova scrittura

```
CREATE OR REPLACE FUNCTION
cantiere.data_nuova_scrittura()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    ultima_scrittura DATE := NULL;
BEGIN
    SELECT v.data_scrittura INTO ultima_scrittura
    FROM cantiere.valore AS v
    WHERE v.valore_di = NEW.valore_di
    ORDER BY v.data_scrittura DESC
    LIMIT 1;

    IF ultima_scrittura >= NEW.data_scrittura
    THEN
        RAISE EXCEPTION 'impossibile inserire una scrittura meno recente';
    ELSE
        RETURN NEW;
```

```
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER data_nuova_scrittura_trig
BEFORE INSERT OR UPDATE ON cantiere.valore
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.data_nuova_scrittura();
```

- Vincolo amministratore delega

```
CREATE OR REPLACE FUNCTION
cantiere.amministratore_corretto_delega()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    amministratore_id INTEGER;
BEGIN
    SELECT c.aperto_da INTO amministratore_id
    FROM cantiere.cantiere AS c
    WHERE c.id = NEW.cantiere_id;
    IF amministratore_id <> NEW.amministratore_id
    THEN
        RAISE EXCEPTION 'il cantiere é stato aperto da un altro amministratore';
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER amministratore_corretto_delega_trig
BEFORE INSERT OR UPDATE ON cantiere.delega
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.amministratore_corretto_delega();
```

- Vincolo solo operatore installa sensore

```
CREATE OR REPLACE FUNCTION
cantiere.solo_operatore_installa_sensore()
```

```
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    ruolo cantiere.RUOLO;
BEGIN
    SELECT o.ruolo INTO ruolo
    FROM cantiere.operaio AS o
    WHERE o.id = NEW.installato_da;
    IF ruolo <> 'operatore'
    THEN
        RAISE EXCEPTION '% non é un operatore', NEW.installato_da;
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER solo_operatore_installa_sensore_trig
BEFORE INSERT OR UPDATE ON cantiere.sensore
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.solo_operatore_installa_sensore();
```

- Vincolo numero civico naturale

```
CREATE OR REPLACE FUNCTION
cantiere.numero_civico_naturale()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
    IF NEW.numero_civico <= 0
    THEN
        RAISE EXCEPTION 'Il numero civico deve essere positivo';
    ELSE
        RETURN NEW;
    END IF;
END;
$$
```

```
CREATE OR REPLACE TRIGGER solo_operatore_installa_sensore_trig
BEFORE INSERT OR UPDATE ON cantiere.luogo
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.numero_civico_naturale();
```

- Vincolo max due sensori differenti

```
CREATE OR REPLACE FUNCTION
cantiere.max_due_sensori_differenti()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    already_installed RECORD := NULL;
BEGIN
    SELECT * INTO already_installed
    FROM cantiere.sensore AS s
    WHERE s.installato_in = NEW.installato_in AND s.tipo = NEW.tipo;
    IF already_installed <> NULL
    THEN
        RAISE EXCEPTION 'sensore di tipo % giá installato in %', NEW.tipo, NEW.ins
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER max_due_sensori_differenti_trig
BEFORE INSERT OR UPDATE ON cantiere.sensore
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.max_due_sensori_differenti();
```

- Vincolo $\text{CAP}_{\text{ha5caratteri}}$

```
CREATE OR REPLACE FUNCTION
cantiere.CAP_ha_5_caratteri()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
```

```
DECLARE
BEGIN
    IF LENGTH(NEW.CAP) <> 5
    THEN
        RAISE EXCEPTION 'Il CAP deve essere composto di 5 cifre, [%] non é valido'
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER CAP_ha_5_caratteri_trig
BEFORE INSERT OR UPDATE ON cantiere.luogo
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.CAP_ha_5_caratteri();
```

- Vincolo CAP composto solo da cifre

```
CREATE OR REPLACE FUNCTION
cantiere.CAP_composto_solo_da_cifre()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
    IF LENGTH(REGEXP_REPLACE(NEW.CAP, '[^0-9]', '', 'g')) <> 5
    THEN
        RAISE EXCEPTION '[%] CAP non valido, contiene valori non numerici', NEW.CA
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER CAP_composto_solo_da_cifre_trig
BEFORE INSERT OR UPDATE ON cantiere.luogo
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.CAP_composto_solo_da_cifre();
```

- Vincolo luogo univoco

```
CREATE OR REPLACE FUNCTION
cantiere.luogo_univoco()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    luogo RECORD := NULL;
BEGIN
    SELECT * INTO luogo
    FROM cantiere.luogo AS l
    WHERE l.CAP = NEW.CAP AND
          l.indirizzo = NEW.indirizzo AND
          l.numero_civico = NEW.numero_civico AND
          l.citta = new.citta;

    IF luogo <> NULL
    THEN
        RAISE EXCEPTION 'Luogo giá presente';
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER luogo_univoco_trig
BEFORE INSERT OR UPDATE ON cantiere.luogo
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.luogo_univoco();
```

- Vincolo area univoca cantiere

```
CREATE OR REPLACE FUNCTION
cantiere.area_univoca_per_cantiere()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    area RECORD := NULL;
BEGIN
    SELECT * INTO area
```

```
    FROM cantiere.area AS a
    WHERE a.tipo = NEW.tipo;
    IF area <> NULL
    THEN
        RAISE EXCEPTION 'Area giá presente nel cantiere';
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER area_univoca_per_cantiere_trig
BEFORE INSERT OR UPDATE ON cantiere.area
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.area_univoca_per_cantiere();
```

- Vincolo valore scritto positivo

```
CREATE OR REPLACE FUNCTION
cantiere.valore_scritto_positivo()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
    IF NEW.valore < 0
    THEN
        RAISE EXCEPTION 'Il valore da associare ad una scrittura deve essere posit
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER valore_scritto_positivo_trig
BEFORE INSERT OR UPDATE ON cantiere.valore
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.valore_scritto_positivo();
```

- Vincolo data scrittura dopo inizio lavori

```
CREATE OR REPLACE FUNCTION
cantiere.data_scrittura_dopo_inizio_lavori()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    sensore RECORD := NULL;
    inizio_lavori TIMESTAMP := NULL;
BEGIN
    SELECT c.data_inizio::timestamp INTO inizio_lavori
    FROM cantiere.cantiere AS C
    WHERE c.id = (SELECT a.parte_di
                  FROM cantiere.area AS a
                  WHERE  a.id = (
                         SELECT s.installato_in
                         FROM cantiere.sensore AS s
                         WHERE s.id = NEW.valore_di));

    IF NEW.data_scrittura <= inizio_lavori
    THEN
        RAISE EXCEPTION 'La lettura é antecedente ai lavori.';
    ELSE
        RETURN NEW;
    END IF;
END;
$$

CREATE OR REPLACE TRIGGER valore_scritto_positivo_trig
BEFORE INSERT OR UPDATE ON cantiere.valore
FOR EACH ROW
    EXECUTE PROCEDURE cantiere.data_scrittura_dopo_inizio_lavori();
```

Popolazione della base dati

Operaio

```
INSERT INTO cantiere.operaio(nome, cognome, ruolo) VALUES
('Mario', 'Rossi', 'semplice'),
('Pasquale', 'Esposito', 'idraulico'),
('Ciro', 'Esposito', 'macchinista'),
```

```
('Jose', 'Miranda', 'elettricista'),
('Chief', 'Braga', 'operatore'),

('Mario', 'Verdi', 'semplice'),
('René', 'Ferretti', 'idraulico'),
('Giuseppe', 'Verdi', 'macchinista'),
('Valerio', 'Brida', 'elettricista'),
('Michele', 'Sua', 'operatore'),

('Sergio', 'Lang', 'semplice'),
('Alessandro', 'Roma', 'idraulico'),
('Domenico', 'Bini', 'macchinista'),
('Mirco', 'Sorrentino', 'elettricista'),
('Antonio', 'Petrillo', 'operatore'),

('Gerardo', 'Pujaz', 'semplice'),
('Bernardo', 'Lico', 'idraulico'),
('Christian', 'Ice', 'macchinista'),
('Matteo', 'Montesi', 'elettricista'),
('Gerardo', 'Malanga', 'operatore');
```

Capocantiere

```
INSERT INTO cantiere.capocantiere(nome, cognome) VALUES
('Francesco', 'Petrillo'),
('Domenico', 'Petrillo'),
('Manuel', 'Scarpitta');
```

Amministratore

```
INSERT INTO cantiere.amministratore(nome, cognome) VALUES
('Mr', 'Implenia'),
('Richard', 'Benson');
```

Luogo

```
INSERT INTO cantiere.luogo(citta, cap, indirizzo, numero_civico) VALUES
('San Giovanni a Piro','84070', 'Via Iacine', 3),
('Fuorigrotta', '80125', 'Via Mercantini', 10),
('Montreaux', '19200', 'Rue du Guercet', 5);
```

Cantiere

```
INSERT INTO cantiere.cantiere(data_inizio, data_fine_prevista, luogo_id, aperto_da) VAI
('2019-07-17', '2019-08-16', 4, 1),
('2000-02-09', '2005-06-1', 2, 2),
('2023-07-1', '2023-08-30', 3, 1);
```

Area

```
INSERT INTO cantiere.area(zona, parte_di, controllato_da) VALUES
('servizi_pubblici', 5, 15),
('zona_verde', 5, 15),
('zona_residenziale', 5, 18),
('zona_ristorazione', 5, 18),
('servizi_pubblici', 4, 13),
('zona_verde', 4, 20),
('servizi_pubblici', 6, 7),
('zona_ristorazione', 6, 5),
('zona_verde', 6, 11);
```

Sensore

```
INSERT INTO cantiere.sensore(tipo, installato_da, installato_in, soglia) VALUES
('gas', 15, 1, 200),
('rumore', 15, 1, 300),
('gas', 15, 2, 200),
('rumore', 15, 2, 300),
('gas', 15, 3, 200),
('rumore', 15, 3, 300),
('gas', 15, 4, 200),
('rumore', 15, 4, 300),
('gas', 5, 5, 400),
('gas', 10, 6, 300),
('rumore', 10, 6, 100),
('rumore', 20, 7, 800),
('gas', 20, 8, 250),
('rumore', 20, 9, 100);
```

Valore

Sensore 1

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 1, 150),
('2000-05-12 01:00:00-07', 1, 50),
('2001-05-12 02:00:00-07', 1, 75),
('2001-05-12 03:00:00-07', 1, 95),
('2002-05-12 04:00:00-07', 1, 100),
('2002-05-12 05:00:00-07', 1, 50),
('2003-05-12 06:00:00-07', 1, 28),
('2003-05-12 07:00:00-07', 1, 50),
('2004-05-12 08:00:00-07', 1, 60);
```

Sensore 2

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 2, 150),
('2000-05-12 01:00:00-07', 2, 50),
('2001-05-12 02:00:00-07', 2, 75),
('2001-05-12 03:00:00-07', 2, 95),
('2002-05-12 04:00:00-07', 2, 100),
('2002-05-12 05:00:00-07', 2, 50),
('2003-05-12 06:00:00-07', 2, 28),
('2003-05-12 07:00:00-07', 2, 50),
('2004-05-12 08:00:00-07', 2, 60);
```

Sensore 3

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 3, 150),
('2000-05-12 01:00:00-07', 3, 50),
('2001-05-12 02:00:00-07', 3, 75),
('2001-05-12 03:00:00-07', 3, 95),
('2002-05-12 04:00:00-07', 3, 100),
('2002-05-12 05:00:00-07', 3, 50),
('2003-05-12 06:00:00-07', 3, 28),
('2003-05-12 07:00:00-07', 3, 50),
('2004-05-12 08:00:00-07', 3, 60);
```

Sensore 4

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 4, 150),
('2000-05-12 01:00:00-07', 4, 50),
('2001-05-12 02:00:00-07', 4, 75),
('2001-05-12 03:00:00-07', 4, 95),
('2002-05-12 04:00:00-07', 4, 100),
('2002-05-12 05:00:00-07', 4, 50),
('2003-05-12 06:00:00-07', 4, 28),
('2003-05-12 07:00:00-07', 4, 50),
('2004-05-12 08:00:00-07', 4, 60);
```

Sensore 5

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 5, 150),
('2000-05-12 01:00:00-07', 5, 50),
('2001-05-12 02:00:00-07', 5, 75),
('2001-05-12 03:00:00-07', 5, 95),
('2002-05-12 04:00:00-07', 5, 100),
('2002-05-12 05:00:00-07', 5, 50),
('2003-05-12 06:00:00-07', 5, 28),
('2003-05-12 07:00:00-07', 5, 50),
('2004-05-12 08:00:00-07', 5, 60);
```

Sensore 6

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 6, 150),
('2000-05-12 01:00:00-07', 6, 50),
('2001-05-12 02:00:00-07', 6, 75),
('2001-05-12 03:00:00-07', 6, 95),
('2002-05-12 04:00:00-07', 6, 100),
('2002-05-12 05:00:00-07', 6, 50),
('2003-05-12 06:00:00-07', 6, 28),
('2003-05-12 07:00:00-07', 6, 50),
('2004-05-12 08:00:00-07', 6, 60);
```

Sensore 7

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 7, 150),
('2000-05-12 01:00:00-07', 7, 50),
('2001-05-12 02:00:00-07', 7, 75);
```

Sensore 8

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 8, 150),
('2000-05-12 01:00:00-07', 8, 50),
('2001-05-12 02:00:00-07', 8, 75);
```

Sensore 9

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 9, 150),
('2000-05-12 01:00:00-07', 9, 50),
('2001-05-12 02:00:00-07', 9, 75);
```

Sensore 10

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 10, 150),
('2000-05-12 01:00:00-07', 10, 50),
('2001-05-12 02:00:00-07', 10, 75);
```

Sensore 11

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 11, 150),
('2000-05-12 01:00:00-07', 11, 50),
('2001-05-12 02:00:00-07', 11, 75);
```

Sensore 12

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 12, 150),
('2000-05-12 01:00:00-07', 12, 50),
('2001-05-12 02:00:00-07', 12, 75);
```

Sensore 13

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 13, 150),
('2000-05-12 01:00:00-07', 13, 50),
('2001-05-12 02:00:00-07', 13, 75);
```

Sensore 14

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 14, 150),
('2000-05-12 01:00:00-07', 14, 50),
('2001-05-12 02:00:00-07', 14, 75);
```

Sensore 15

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 15, 150),
('2000-05-12 01:00:00-07', 15, 50),
('2001-05-12 02:00:00-07', 15, 75);
```

Sensore 16

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 16, 150),
('2000-05-12 01:00:00-07', 16, 50),
('2001-05-12 02:00:00-07', 16, 75);
```

Sensore 17

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 17, 150),
('2000-05-12 01:00:00-07', 17, 50),
('2001-05-12 02:00:00-07', 17, 75);
```

Sensore 18

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 18, 150),
('2000-05-12 01:00:00-07', 18, 50),
('2001-05-12 02:00:00-07', 18, 75);
```

Sensore 19

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 19, 150),
('2000-05-12 01:00:00-07', 19, 50),
('2001-05-12 02:00:00-07', 19, 75);
```

Sensore 20

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 20, 150),
('2000-05-12 01:00:00-07', 20, 50),
('2001-05-12 02:00:00-07', 20, 75);
```

Sensore 21

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 21, 150),
('2000-05-12 01:00:00-07', 21, 50),
('2001-05-12 02:00:00-07', 21, 75);
```

Sensore 22

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 22, 150),
('2000-05-12 01:00:00-07', 22, 50),
('2001-05-12 02:00:00-07', 22, 75);
```

Sensore 23

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 23, 150),
('2000-05-12 01:00:00-07', 23, 50),
('2001-05-12 02:00:00-07', 23, 75);
```

Sensore 24

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 24, 150),
('2000-05-12 01:00:00-07', 24, 50),
('2001-05-12 02:00:00-07', 24, 75);
```

Sensore 25

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 25, 100),
('2000-05-12 01:00:00-07', 25, 50),
('2001-05-12 02:00:00-07', 25, 75);
```

Sensore 26

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 26, 100),
('2000-05-12 01:00:00-07', 26, 50),
('2001-05-12 02:00:00-07', 26, 75);
```

Sensore 27

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 27, 100),
('2000-05-12 01:00:00-07', 27, 50),
('2001-05-12 02:00:00-07', 27, 75);
```

Sensore 28

```
INSERT INTO cantiere.valore(data_scrittura, valore_di, valore) VALUES
('2000-05-12 00:00:00-07', 28, 100),
('2000-05-12 01:00:00-07', 28, 50),
('2001-05-12 02:00:00-07', 28, 75);
```

Delega

```
INSERT INTO cantiere.delega(cantiere_id, amministratore_id, capocantiere_id) VALUES
(4,1,1),
(5,2,2),
(6,1,3);
```

Lavora in

```
INSERT INTO cantiere.lavora_in(cantiere_id, operaio_id) VALUES
(4,1),
(4,2),
(4,3),
(4,4),
```

```
(4,5),
(4,6),
(4,7),
(4,8),
(5,9),
(5,10),
(5,11),
(5,12),
(5,13),
(5,14),
(5,15),
(5,16),
(5,17),
(6,18),
(6,19),
(6,20);
```

Extra

File docker compose

Di seguito la configurazione di `Docker compose` per poter utilizzare la base dati.

```
version: "3.9"
services:
  postgres:
    container_name: pg
    restart: always
    image: postgres
    environment:
      POSTGRES_PASSWORD: postgres
      POSTGRES_USER: postgres
      POSTGRES_DB: cantiere
    ports:
      - "5432:5432"
  pgadmin:
    container_name: pg_admin
    image: dpage/pgadmin4
    restart: always
```

```
environment:
  PGADMIN_DEFAULT_EMAIL: antonio.petrillo4@studenti.unina.it
  PGADMIN_DEFAULT_PASSWORD: root
ports:
  - "5050:80"
```

Set search path

Con la seguente query si puó settare il `search path` del DBMS in modo tale che non sia necessario aggiungere il nome dello schema come prefisso, in questo modo é possibile scrivere delle query piú brevi. In generale non é una buona abitudine da utilizzare su un database in produzione.

```
SET search_path TO cantiere;
```