TESINA BASI DI DATI – A.A. 2014/2015

Francesco Bellei, Bruno Ghion

a) Interpretazione e completamento dei requisiti

- Testo originale (con entità segnate):

Si vuole gestire un database per registrare le informazioni relative al magazzino di una ditta di fresatura, tornitura e piccola carpenteria.

MAGAZZINO

Le materie prime sono da intendersi come profilati di varie misure, ognuno dei quali è fornito da più fornitori. Di ogni materia prima si conoscono la descrizione, il materiale (Ferro, Bronzo, Ottone, …), la categoria (Piatto, Tondo, …), lo stato del materiale (Trafilato, Rettificato, Laminato, …), l'unità di misura (tipicamente il kg o il metro), la quantità presente in magazzino. Una materia prima può essere fornita da più fornitori, ciascuno dei quali offre la merce a un prezzo diverso variabile nel tempo e con tempi di consegna differenti. Alcune delle merci sono tenute "a magazzino" ovvero il loro quantitativo totale in deposito non deve mai scendere al di sotto di una soglia minima fissata. Nel caso in cui ciò non avvenga, l'operatore emette un ordine di nuovo materiale richiedendolo ad uno tra i fornitori disponibili. Occorre prevedere di poter gestire il materiale impegnato, ovvero la quantità destinata ad ordini accettati, ma non ancora in esecuzione. La lavorazione avviene invece solo su commissione e dunque non esiste un magazzino di pezzi già lavorati.

ARTICOLI PRODOTTI

L'applicazione deve tenere conto della traccia di tutti gli articoli prodotti. Per ogni articolo si conosce il codice dell'articolo, il cliente che lo richiede, il costo di produzione, il prezzo di vendita, il tempo di produzione, la quantità da produrre relativamente ad un dato ordine, la descrizione, l'elenco dei materiali componenti e la quantità necessaria di ciascuno di essi. Ogni articolo è sottoposto ad un "ciclo di lavoro".

CICLO DI LAVORO

Ogni articolo ha un "ciclo di lavoro" unico che consiste in una sequenza di fasi lavorative (operazioni) ciascuna con una durata ed un numero progressivo. Il ciclo è caratterizzato da un tempo di lavorazione totale, un tempo di preparazione che tiene conto della somma dei tempo di preparazione degli utensili per effettuare le diverse lavorazioni sui pezzi (siano essi semplici o composti) componenti e da un tempo per la predisposizione del ciclo detto "tempo di pulizia". Le operazioni fatte sulle materie prime sono eseguite da macchine identificate fa un codice univoco.

GESTIONE MACCHINE

Le macchine che eseguono le operazioni possono essere oggetto di manutenzione, che può essere di due tipi: interna, effettuata da un operaio, quindi di carattere ordinario come il cambio di batteria; esterna, eseguita da una ditta che comporta l'uso di pezzi di ricambio. Deve essere possibile inserire nuovi macchinari o cancellare macchinari non più utilizzati.

GESTIONE DEGLI ORDINI (effettuati dai clienti)
Un ordine viene effettuato da un cliente ed è composto dalla richiesta di diversi quantitativi di vari articoli.
Un ordine può avere tre stati consecutivi: **accettato** (lo stato di un ordine che è posto nella coda degli ordini da eseguire; le materie prime ad esso relative devono risultare come impegnate), **in esecuzione** (lo stato di un ordine i cui articoli sono in fase di lavorazione) e **terminato** (tutti gli articoli relativi a tale ordine sono stati eseguiti). Una volta che un ordine entra in esecuzione, e cioè anche un solo articolo è stato iniziato, è necessario aggiornare i quantitativi di materie prime già presenti in magazzino necessarie per soddisfarlo.

- Interpretazione del testo:

- Una **materia prima** è identificata da un codice e descritta da una descrizione, dal materiale, dalla categoria, dallo stato del materiale, dall'unità di misura, la quantità presente nel magazzino, la soglia della quantità (che se viene superata, viene effettuato un ordine), e da una quantità impegnata per gli ordini. Una materia prima può essere usata per creare un articolo, è fornita da diversi fornitori.
- Una **persona** è identificata dal codice fiscale e descritta dal nome e dal cognome.
- Una persona può essere un **fornitore**, descritto anche dalla descrizione, dal tempo di consegna e dal prezzo del prodotto, questo fornisce una sola materia prima, ad esso vengono richiesti gli ordini per le materie prime.
- Una persona può essere un **operatore** che può effettuare degli ordini di materie prime.
- Una persona può essere un **cliente** che può effettuare degli ordini di articoli.
- Una persona può essere un **operaio** che effettua una manutenzione interna.
- Una **manutenzione** è identificata da un codice univoco e dal codice della macchina, è descritto da una data e da una descrizione. Viene eseguita su una macchina.
- Una manutenzione può essere **interna** ed è identificata anche dal codice dell'operaio che la effettua.

- Una manutenzione può essere **esterna** ed è descritta anche dal nome del pezzo di ricambio utilizzato e dalla ditta.
- Una **macchina** è identificata dal codice univoco e descritta da una descrizione e da un valore che indica se essa è in uso oppure no. Una macchina può subire delle manutenzioni e eseguire delle operazioni.
- Un'**operazione** è identificata da un numero progressivo e dal codice del ciclo, è descritta dalla sua durata e da una descrizione. Un'operazione viene eseguita da una macchina, più operazioni compongono un ciclo.
- Un **ciclo** è identificato da un codice identificativo e dal codice dell'articolo,è descritto da un tempo di lavorazione totale, da un tempo di preparazione che tiene conto della somma del tempo di preparazione degli utensili e da un tempo di pulizia. Il ciclo è composto da diverse operazioni ed è posseduto da un articolo.
- Un **utensile** è identificato da un codice e descritto dal nome e dal tempo di preparazione. Può essere posseduto da un ciclo, questa opzionalità è data dal fatto che un utensile può essere utilizzato fino a fine ciclo, ma ancora in buono stato quando il ciclo viene rimosso.
- Un **articolo** è identificato da un codice e descritto da una descrizione, dal costo di produzione, dal prezzo di vendita, dal tempo di produzione. Un articolo è ordinato una o più volte, nella relazione con l'ordine viene descritta la quantità di articoli ordinati. Un articolo necessita di uno o più materie prime e nella relazione viene indicata la quantità di materie prime necessarie. Un articolo ha un ciclo di lavoro.
- Un **ordine dell'operatore** è identificato dal codice dell'ordine e dal codice fiscale dell'operatore che lo effettua, è descritto dalla data in cui viene effettuato. Un ordine di materie prime è associato con uno o più fornitori.
- Un **ordine del cliente** è identificato da un codice e dal codice fiscale del cliente che lo effettua, è descritto dalla data, dalla quantità totale di articoli, ha una descrizione e da uno stato. Un ordine degli articoli richiede uno o più articoli, nella relazione è indicata anche la quantità dell'articolo associato. Un ordine degli articoli è effettuato da un solo cliente.

- Glossario:

TERMINE	DESCRIZIONE	SINONIMI	LEGAME

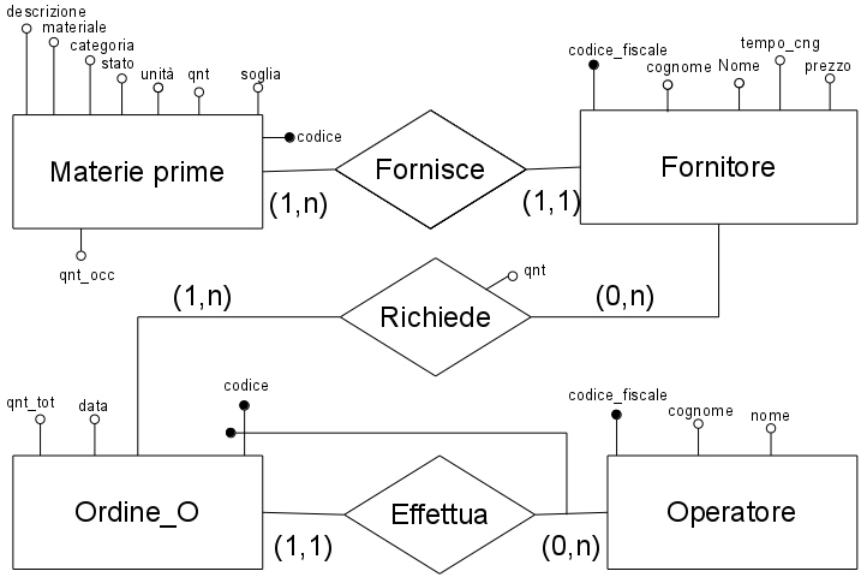
Materia prima	Descrizione Materiale Categoria Stato materiale Unità di misura Quantità Soglia Quantità	Merce Materiale	Fornitore Articolo
Fornitore	Codice fiscale Nome Cognome Descrizione prezzo tempo_cng		Materia prima Ordine degli operatori
Operatore	Codice fiscale Nome Cognome		Ordine degli operatori
Cliente	Codice fiscale Nome Cognome		Ordine dei clienti
Operaio	Codice fiscale Nome Cognome		Manutenzione
Manutenzione	Data Descrizione		Macchina Operaio
Macchina	Codice univoco Descrizione	Macchinario	Manutenzione Operazione
Operazione	Numero progressivo Durata Descrizione	Fase lavorativa	Ciclo Macchina
Ciclo	Tempo totale Tempo di preparazione Tempo di pulizia	Ciclo di lavoro	Operazione Articolo
Articolo	Nome Descrizione Costo di produzione Prezzo di vendita Tempo di produzione		Materia prima Ordine del cliente Ciclo

Ordine dell'operatore	Data Stato		Fornitore
Ordine del cliente	Data Stato Descrizione Quantità articoli		Articolo Cliente

b) Progetto concettuale – Schema E/R

Di seguito le fasi che conducono allo schema E/R finale:

Prima Fase (Magazzino):



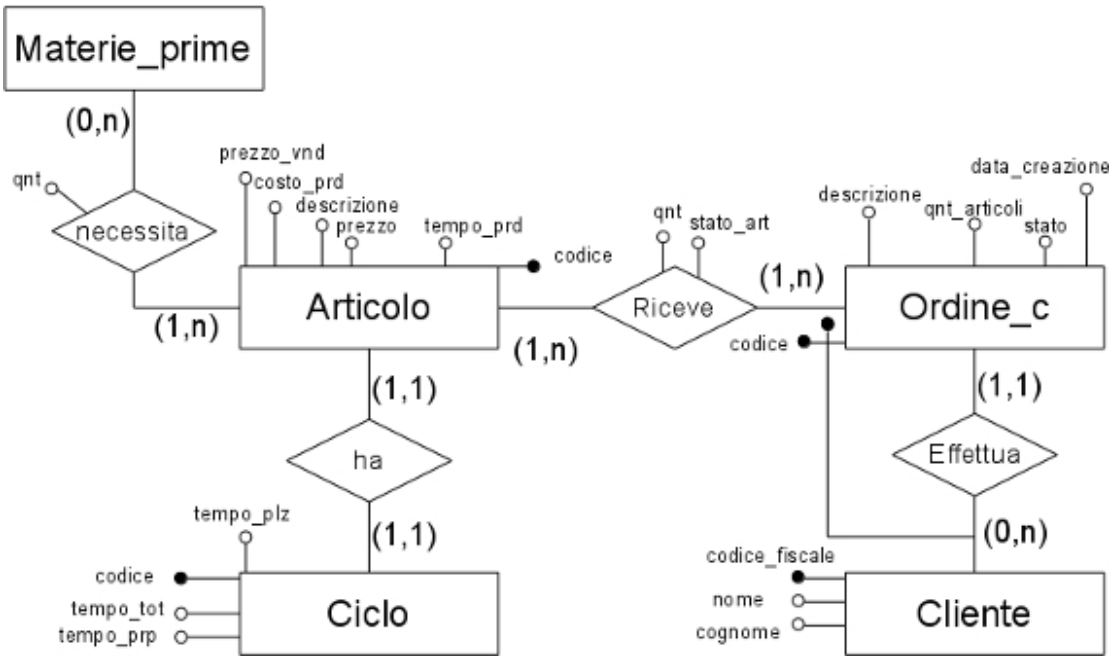
Il primo paragrafo del testo suggerisce 4 entità: Materie prime, Fornitori, Ordine e Operatore. Ogni Materia prima è fornita da più fornitori, non è specificato però se un fornitore può fornire più materie prime, per semplificazione assumiamo che un fornitore forniesce solo una materia prima con una quantità chiesta al momento di scrittura di un ordine. Il prezzo e il tempo di consegna sono contenute nell'entità fornitore poiché solo un fornitore è legato a una materia prima. Ogni materia prima ha la propria soglia minima sotto la quale non può scendere, se al contrario, il valore della soglia non è impostato, la soglia è impostata a 0.

Un operatore può effettuare più ordini della stessa materia prima mentre è effettuato da un solo operatore, un ordine non può essere emesso da più operatori. Un ordine esiste solo se esiste un operatore che lo ha emesso.

Ogni materia prima ha una quantità (qnt) presente in magazzino libera e disponibile per la creazione di articoli e una quantità occupata (qnt_occ) per ordini accettati ma non ancora in

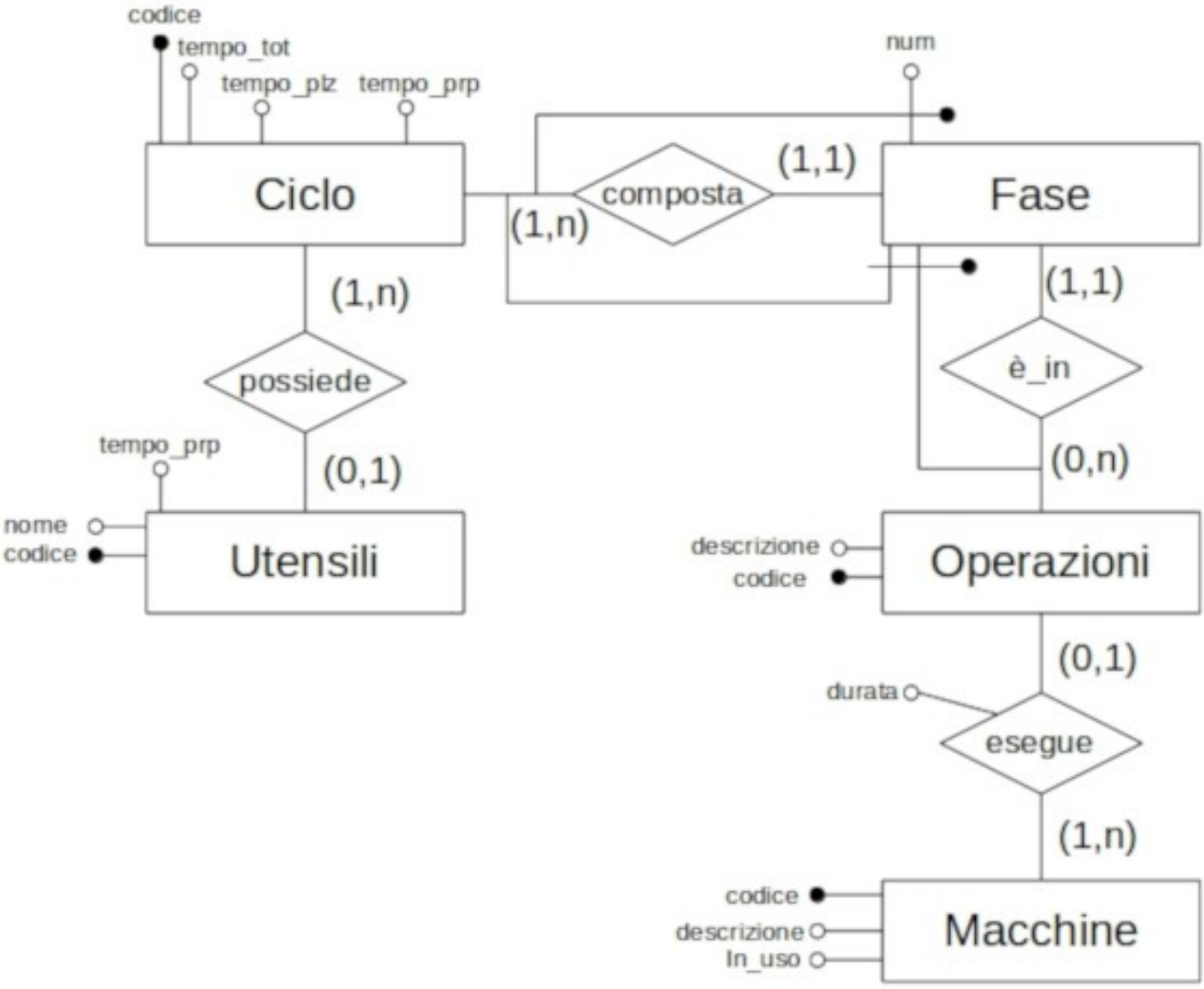
esecuzione.
Per gestire il materiale impegnato, ovvero la quantità ad ordini accettati, ma non ancora in esecuzione utilizziamo l'attributo stato in Ordine_C (seconda fase).

Seconda Fase (Articoli prodotti e Gestione ordini):



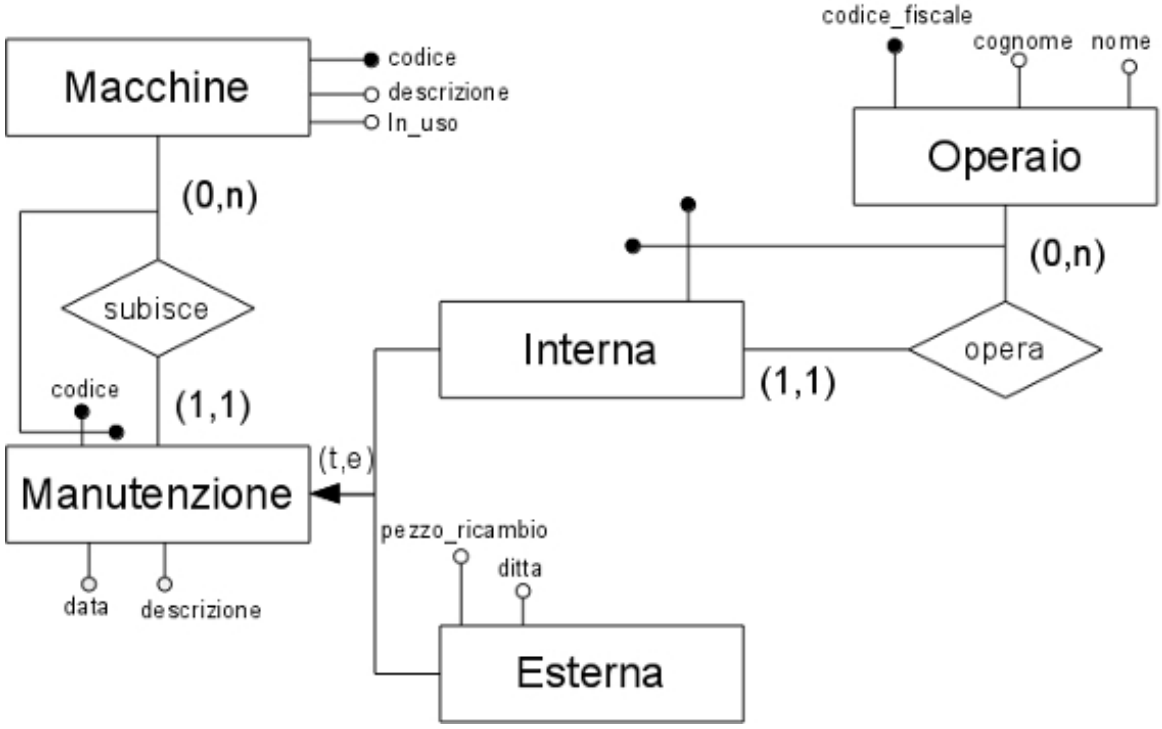
Il secondo paragrafo è composto da 4 nuove entità: articolo, ordine, cliente e ciclo.
Ogni cliente può effettuare uno o più ordini, ogni ordine è effettuato da un solo cliente. Un ordine esiste solo se esiste un cliente che lo ha effettuato. Ogni ordine è composto da almeno un articolo. Lo stato di un'ordine può essere: “non accettato”, “accettato”, “in esecuzione”, “terminato”. A livello software aggiorneremo la quantità di materie prime occupate e disponibili ogni volta che un ordine passa dallo stato “accettato” allo stato “In esecuzione”. Ogni ordine ha una quantità totale di articoli ordinati (qnt_articoli) che essendo un possibile attributo derivato discuteremo in seguito.
Ogni articolo necessita una certa quantità per ogni materia prima che lo compone.
Ogni articolo ha un proprio ciclo di lavoro dal quale viene creato.

Terza Fase (Ciclo di lavoro):



Entità: Ciclo, Fase, Operazioni, Utensili, Macchine.
Ogni articolo ha un solo ciclo e ogni ciclo ha un solo articolo associato. Ogni ciclo è associato a n fasi ognuna descritta dal numero che identifica la fase a quel ciclo, (es il Ciclo A001 ha 3 Fasi: num 1, num 2, num 3). Si ha la possibilità di avere in cicli diversi una fase uguale, non si può avere la stessa fase più volte nello stesso ciclo. Un ciclo di lavoro può avere associate più operazioni diverse tra di loro, ma non uguali.
Si è preferito creare due entità separate, Operazioni e Fase, in modo da non avere tuple ripetute in operazioni, scenario che si realizzerebbe se operazioni e fase fossero un attributo di operazioni.
Il tempo totale del ciclo è un possibile dato derivato che in seguito discuteremo.

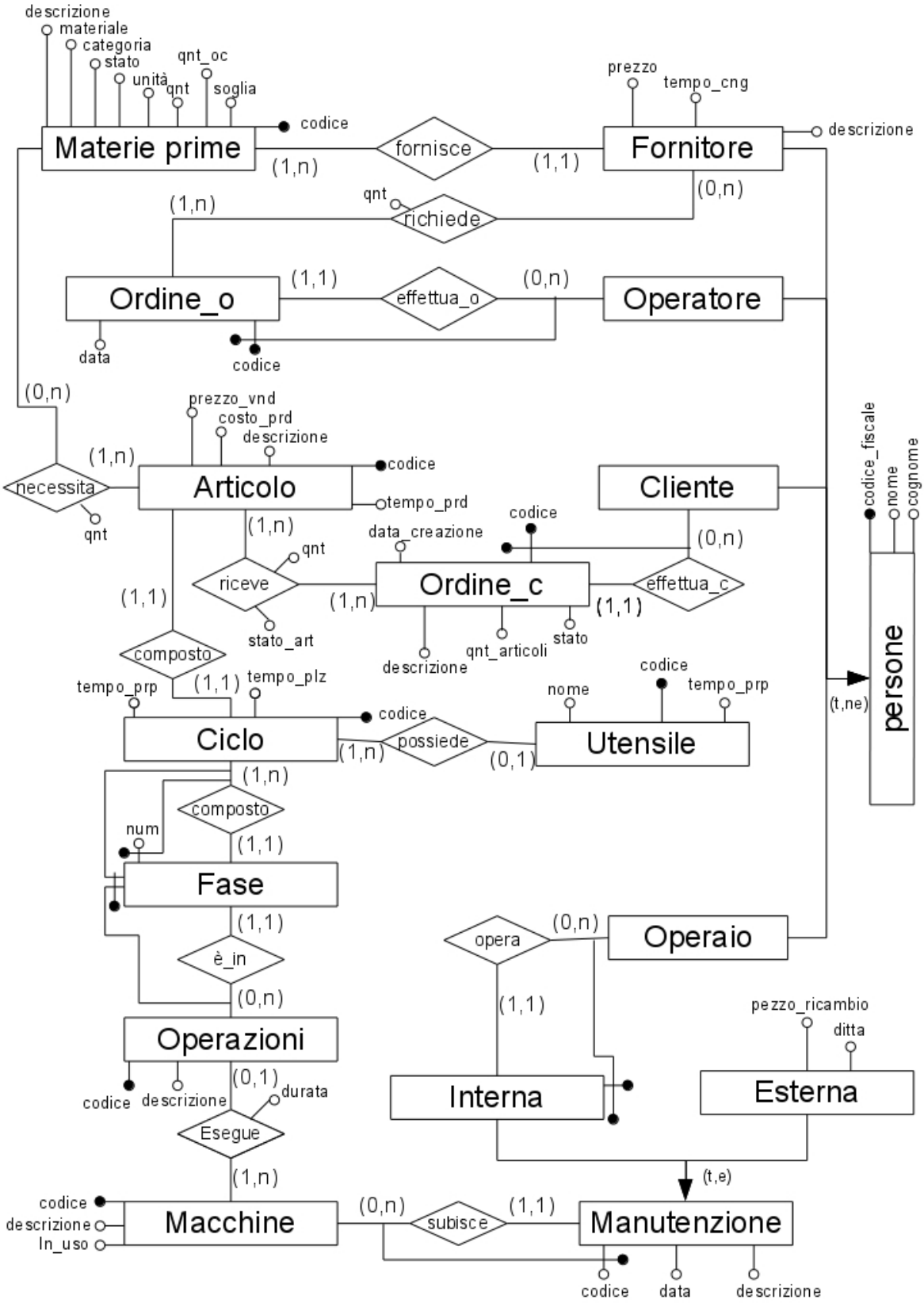
Quarta Fase (Macchine):



Entità: Macchine, Manutenzione, Interna, Esterna, Operaio.
La creazione della gerarchia Manutenzione permette di avere gli stessi attributi (data e descrizione) indipendentemente se sia stata effettuata da un operaio interno o esterno alla ditta, nella manutenzione, nel caso in cui sia interna, sarà obbligatorio indicare l'operaio che lo ha effettuato, mentre per manutenzioni esterne basterà indicare la ditta e il pezzo di ricambio.
Quando una macchina viene sostituita da un'altra, il sistema non registra questo cambiamento, la vecchia macchina viene eliminata e viene registrata la nuova.
La possibilità dell'eliminazione delle macchine comporta possibili “buchi” con le operazioni, ovvero c'è la possibilità che esista una operazione ma non sia eseguita da nessuna macchina, di conseguenza ad un articolo mancherebbe l'operazione associata. La possibilità di creare operazioni senza una macchina associata lascia più libertà agli utenti che utilizzeranno l'applicazione.

Schema complessivo:

Nello



schema complessivo compare una nuova entità persona in modo da creare una gerarchia tra le varie entità che hanno in comune gli stessi attributi.

Inoltre Macchine e Operazioni sono legati da una relazione (1,n) che contiene la durata dell'operazione, poiché nel caso in cui venga sostituita una macchina la durata di una operazione può essere diversa.

c) Schema relazionale – Progetto logico

Materie_Prime (codice, descrizione, materiale, categoria, stato, unità, qnt, qnt_occ, soglia)

Fornitore(codice_fiscale, codice_mat, nome, cognome,descrizione,prezzo,tempo_cng)
FK: codice_mat **References** Materie_Prime

Richiede (codice_for, codice_ord_o, qnt)
FK: codice_mat **References** Fornitore
FK: codice_ordine_o **References** Ordine_O

Ordine_O (codice, codice_opr, data)
FK: codice_opr **References** Operatore

Operatore (codice_fiscale, nome, cognome)

Articolo (codice, descrizione, costo_prd, prezzo_vnd, tempo_prd)

Ordine_C (codice, codice_clt, qnt_art, data, stato, descrizione)
FK: codice_clt **References** Cliente

Necessita (codice_mat, codice_art, qnt)
FK: codice_mat **References** Materie_Prime
FK: codice_art **References** Articolo

Riceve (codice_art, codice_ord_c, qnt, stato_art)
FK: codice_art **References** Articolo
FK: codice_ordine_c **References** Ordine_C

Cliente (codice_fiscale, nome, cognome)

Ciclo (codice, codice_art, tempo_prp, tempo_plz)
FK: codice_art **References** Articolo

Utensile (codice, codice_ccl, nome, tempo_prp)
FK: codice_ccl **References** Ciclo

Operazioni (codice, descrizione)

Esegue (codice_operazioni, codice_mac, durata)
FK: codice_operazioni **References** Operazioni

Fase (num, codice_ccl, codice_opr)
- **AK**: codice_ccl, codice_opr
- **FK**: codice_ccl **References** Ciclo
- **FK**: codice_opr **References** Operazioni

Macchine (codice, descrizione, in_uso)

Manutenzione (codice, codice_mac, data, descrizione)
- **FK**: codice_mac **References** Macchine

Interna (codice, codice_mac, codice_operaio)
- **FK**: codice, codice_mac **References** Manutenzione
- **FK**: codice_operaio **References** Operaio

Esterna (codice, codice_mac, pezzo_ricambio, ditta)
- **FK**: codice, codice_mac **References** Manutenzione

Operaio (codice_fiscale, nome, cognome)

È stato effettuato un collasso verso il basso per la gerarchia di persone poiché non è relazionata con altre entità. Inoltre le entità figlie eseguono operazioni diverse pur avendo gli stessi attributi.

Per la gerarchia di manutenzione, invece, è stato effettuato un mantenimento delle entità, poiché l'entità padre è relazionata con un'altra entità e le entità figlie hanno attributi e relazioni indipendenti.

d) Operazioni previste dalla base di dati

- **Ricerca 1**: Selezionare su base annuale la materia prima tenuta a magazzino più usata per aumentare la soglia.

```
CREATE VIEW Articoli_Quantita AS
(SELECT Riceve.codice_art, SUM(Riceve.qnt) AS quant
 FROM Riceve, Ordine_C
 WHERE Riceve.codice_ord_c = Ordine_C.codice
    AND Ordine_C.data_creazione BETWEEN '2015-01-01' AND '2015-12-31'
 GROUP BY Riceve.codice_art
)

SELECT Materie_Prime.codice AS Materia_prima,
          SUM(Necessita.qnt*Articolo_Quantita.quant) AS Quantita
FROM Materie_Prime, Articolo_Quantita, Necessita
WHERE Necessita.codice_art = Articolo_Quantita.art
    AND Necessita.codice_mat = Materie_Prime.codice
GROUP BY Materie_Prime.codice
ORDER BY Quantita DESC
```

- **Ricerca 2**: Eseguire una stima del tempo impiegato per produrre tutti gli articoli di un certo ordine (per esempio l'ordine con codice 0001):

```
SELECT TIME(SUM(A.tempo_prd*R.qnt)) AS Stima_Tempo
FROM Articolo AS A, Riceve AS R, Ordine_C AS O
WHERE R.codice_art = A.codice
    AND R.codice_ord_c = O.codice
    AND O.codice = '0001';
```

- **Ricerca 3**: Stima di quante manutenzioni sono effettuate su una data macchina, che siano esterne o interne è indifferente.

```
SELECT Count(Man.codice) as Tot_Manutenzioni
FROM Manutenzione as Man, Macchine as Mac
WHERE Man.codice_mac = Mac.codice
AND Mac.codice = '0001';
```

- **Ricerca 4**: Codice dei fornitori che forniscono con il prezzo più basso una data materia prima(per esempio con codice 0002).

```
SELECT f.codice_fiscale AS Codice_Fornitore, MIN(f.prezzo) AS Prezzo
FROM Fornitore AS f
WHERE f.codice_mat = '0002'
```

- **Ricerca 5**: codice delle operazioni che sono contenute in un dato ciclo.

```
SELECT Operazioni.codice
FROM Operazioni, Fase, Ciclo
WHERE Ciclo.codice = Fase.codice_ccl
AND Fase.codice_operazioni = Operazioni.codice
```

AND Ciclo.codice = '0001'

- **Ricerca 6**: Numero di materie prime che compongono un dato articolo (per es con codice 0001).

  SELECT SUM(N.qnt)
  FROM Necessita AS N
  WHERE N.codice_art = '0001'

- **Ricerca 7**: codice degli ordini fatti dai clienti ordinati in base alla data di creazione.
  SELECT Ordine_C.codice
  FROM Ordine_C
  ORDER BY Ordine_C.data_creazione DESC;

- **Ricerca 8**: Gli articoli più venduti.

  SELECT Riceve.codice_art, SUM(Riceve.qnt) AS Quantità
  FROM Riceve
  GROUP BY Riceve.codice_art
  ORDER BY Quantità DESC

- **Ricerca 9**: Codice degli ordini che contengono tutti gli articoli che hanno un prezzo maggiore di 1000.

```
SELECT *
FROM Ordine_C
WHERE NOT EXISTS (
        SELECT *
          FROM Articolo
        WHERE Articolo.prezzo_vnd >= 1000
           AND NOT EXISTS(
               SELECT *
               FROM Riceve
               WHERE Riceve.codice_art = Articolo.codice
               AND Riceve.codice_ord_c = Ordine_C.codice
            )
)
```

Ricerca 10: i codici delle macchine sulle quali sono state effettuate una o più manutenzione ad una certa data (fissata a tempo di scrittura della query per es 0004) e eseguenti una data operazione (per esempio con codice 0001).

```
SELECT Esegue.codice_mac
FROM Esegue
WHERE Esegue.codice_operazioni = '0004'

UNION

SELECT Manutenzione.codice_mac
FROM Manutenzione
WHERE Manutenzione.data = '2015-10-10 00:00:00'
```

- **Eliminazione 1:** Eliminazione della macchina con codice 0004, eiminando le macchine elimino anche le manutenzioni ad esse associate:

  DELETE FROM Macchine WHERE Macchine.codice = '0004'

- **Eliminazione 2:** Eliminazione di un utensile con codice 0001 (un'utensile usurato per esempio):

  DELETE FROM Utensili WHERE Utensili.codice = '0001'

- **Eliminazione 3:** L'eliminazione di un'articolo (codice AB20) comporta l'eliminazione automatica dei cicli e delle fasi ad esso associati:

  DELETE FROM Utensili WHERE Utensili.codice = 'AB20'

- **Eliminazione 4:** L'eliminazione di ciclo (codice 0020) comporta l'eliminazione delle fasi ad esso associati:

  DELETE FROM Ciclo WHERE Ciclo.codice = '0020'

- **Eliminazione 4:** Eliminazione di una fase (codice 0005):

  DELETE FROM Fase WHERE Fase.codice = '0005'

- **Modifica 1:** Modificare lo stato di un dato articolo di un dato ordine effettuato da un cliente (per esempio impostare lo stato dell'articolo di codice '0003' in 'finito'(3), nell'ordine con codice '0001'):

  UPDATE Riceve
  SET stato_art = 2
  WHERE codice_art = '0003'
  AND codice_ord_c = '0001'

- **Modifica 2:** Modificare la quantità di un dato articolo di un dato ordine effettuato da un cliente (per esempio modificare la quantità dell'articolo di codice '0002' a 4, nell'ordine con codice '0004'):

  UPDATE Riceve
  SET qnt = 4
  WHERE codice_art = '0002'
  AND codice_ord_c = '0004'

- **Modifica 3:** Modificare lo stato di un dato ordine effettuato da un cliente (per esempio modificare lo stato di un ordine che ha codice '0003' in 'terminato'(4)):

  UPDATE Ordine_C
  SET stato = 4
  WHERE codice = '0003'

- **Modifica 4:** Modificare la quantità occupata di una data materia prima (per esempio modificare la quantità occupata in 83 unità della materia prima con codice '0005'):

  UPDATE Materie_Prime
  SET qnt_occ = 83
  WHERE codice = '0005'

- **Modifica 5:** Modificare la quantità richiesta di materie prime di un ordine di un operatore (per esempio modificare la quantità richiesta di una materia prima con codice '0001' in 100 unità per l'ordine con codice '0002'):

  UPDATE Richiede
  SET qnt = 100
  WHERE codice_mat = '0001'
  AND codice_ord_o = '0002'

- **Modifica 6:** Modificare l'opzione in_uso delle Macchine (Per esempio, per una macchina con codice '0004' impostare la variabile in uso su 'si'):

  UPDATE Macchine
  SET in_uso = 2
  WHERE codice = '0004'

CREATE TABLE

```
CREATE TABLE `Articolo`
(
 `codice`       CHAR(4) NOTNULL,
 `descrizione` VARCHAR(255) NOT NULL,
 `costo_prd`    INT(10) UNSIGNED NOT NULL,
 `prezzo_vnd`   INT(10) UNSIGNED NOT NULL,
 `tempo_prd`    time NOT NULL DEFAULT '00:00:00',
 PRIMARY KEY (`codice`)
);

CREATE TABLE `Ciclo`
(
 `codice`     CHAR(4) NOT NULL,
 `codice_art` CHAR(4) NOT NULL,
 `tempo_prp`  time NOT NULL DEFAULT '00:00:00',
 `tempo_plz`  time NOT NULL DEFAULT '00:00:00',
 PRIMARY KEY (`codice`),
 FOREIGN KEY(`codice_art`) REFERENCES `articolo`(`codice`) ON DELETE CASCADE ON
UPDATECASCADE
);

CREATE TABLE `Cliente`
(
 `codice_fiscale` CHAR(16) NOT NULL,
 `nome`            VARCHAR(25) NOT NULL,
 `cognome`         VARCHAR(25) NOT NULL,
 PRIMARY KEY (`codice_fiscale`)
);

CREATE TABLE `Esterna`
(
 `codice`          CHAR(4)  NOT NULL,
 `pezzo_ricambio` CHAR(30) NOT NULL,
 `ditta`           CHAR(25) NOT NULL,
 PRIMARY KEY(`codice`),
 FOREIGN KEY(`codice`) REFERENCES `manutenzione`(`codice`) ON DELETE CASCADE ON
 UPDATE CASCADE
);

CREATE TABLE `Fase`
(
 `num`               INT(10) UNSIGNED NOT NULL,
 `codice_operazioni` CHAR(4) NOT NULL,
 `codice_ccl`        CHAR(4) NOT NULL,
 PRIMARY KEY (`num`,`codice_ccl`),
 UNIQUE KEY `codice_operazioni` (`codice_operazioni`,`codice_ccl`),
 FOREIGN KEY(`codice_operazioni`) REFERENCES `operazioni`(`codice`) ON DELETE CASCADE
 ONUPDATE CASCADE,
 FOREIGN KEY(`codice_ccl`) REFERENCES `ciclo`(`codice`) ON DELETE CASCADE
 ON UPDATE CASCADE
);

CREATE TABLE `Fornitore`
(
 `codice_fiscale` CHAR(16) NOT NULL,
 `codice_mat`     CHAR(4) NOT NULL,
 `nome`           VARCHAR(25) DEFAULT NULL,
 `cognome`        VARCHAR(25) DEFAULT NULL,
 `descrizione`    VARCHAR(255)DEFAULT NULL,
 `prezzo`         INT(10) UNSIGNED NOT NULL,
 `tempo_cng`      INT(10) UNSIGNED NOT NULL DEFAULT '1',
```

```
 PRIMARY KEY (`codice_fiscale`),
 FOREIGN KEY (`codice_mat`) REFERENCES `materie_prime` (`codice`) ON DELETE CASCADE
 ON UPDATE CASCADE
);

CREATE TABLE `Interna`
(
 `codice`         CHAR(4) NOT NULL,
 `codice_operaio` CHAR(16) NOT NULL,
 PRIMARY KEY(`codice`,`codice_operaio`),
 FOREIGN KEY(`codice`) REFERENCES `manutenzione`(`codice`) ON DELETE CASCADE
 ON UPDATE CASCADE,
 FOREIGN KEY(`codice_operaio`) REFERENCES `operaio`(`codice_fiscale`) ON DELETE
 CASCADE ON UPDATE CASCADE
);

CREATE TABLE `Macchine`
(
 `codice`      CHAR(4) NOT NULL,
 `descrizione` VARCHAR(255) NOT NULL EFAULT '',
 `in_uso`      ENUM('Si', 'No') NOT NULL DEFAULT 'No',
 PRIMARY KEY (`codice`)
);

CREATE TABLE `manutenzione`
(
 `codice`      CHAR(4) NOT NULL,
 `codice_mac`  CHAR(4) NOT NULL,
 `data`        DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
 `descrizione` VARCHAR(255) NOT NULL DEFAULT '',
 PRIMARY KEY (`codice`,`codice_mac`),
 FOREIGN KEY(`codice_mac`) REFERENCES `macchine`(`codice`) ON DELETE
 CASCADE ON UPDATE CASCADE
);

CREATE TABLE `materie_prime`
(
 `codice`      CHAR(4) NOT NULL,
 `descrizione` VARCHAR(255) DEFAULT NULL,
 `materiale`   VARCHAR(25) DEFAULT NULL,
 `categoria`   VARCHAR(25) DEFAULT NULL,
 `stato`       VARCHAR(25) DEFAULT NULL,
 `unità`       ENUM('Kg', 'm', 'lt')DEFAULT NULL,
 `qnt`         INT(10) UNSIGNED NOT NULL,
 `qnt_occ`     INT(10) UNSIGNED NOT NULL,
 `soglia`      INT(10) UNSIGNED NOT NULL,
 PRIMARY KEY (`codice`)
);

CREATE TABLE `Necessita`
(
 `codice_mat` CHAR(4) NOT NULL,
 `codice_art` CHAR(4) NOT NULL,
 `qnt`        INT(10) UNSIGNED NOT NULL,
 PRIMARY KEY (`codice_mat`,`codice_art`),
 FOREIGN KEY (`codice_mat`) REFERENCES `materie_prime` (`codice`)
 ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`codice_art`) REFERENCES `articolo` (`codice`)
 ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE `Operaio`
(
 `codice_fiscale` CHAR(16) NOT NULL,
 `nome`           CHAR(25) NOT NULL,
 `cognome`        CHAR(25) NOT NULL,
 PRIMARY KEY (`codice_fiscale`)
);

CREATE TABLE `Operazioni`
(
 `codice`      CHAR(4) NOT NULL,
 `descrizione` VARCHAR(255) NOT NULL DEFAULT '',
 PRIMARY KEY(`codice`)
);

CREATE TABLE `Ordine_C`
(
 `codice` CHAR(4) NOT NULL,
 `codice_clt` CHAR(16) NOT NULL,
 `qnt_art` INT(10) UNSIGNED NOT NULL,
 `data_creazione` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
 `stato` ENUM('Non Accettato','Accettato', 'In Esecuzione', 'Terminato')
 NOT NULL DEFAULT 'Non Accettato',
 `descrizione` VARCHAR(255) DEFAULT NULL,
 PRIMARY KEY(`codice`,`codice_clt`),
 FOREIGN KEY(`codice_clt`) REFERENCES `cliente` (`codice_fiscale`)
 ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE `Ordine_O`
(
 `codice`           CHAR(4) NOT NULL,
 `codice_operatore` CHAR(16) NOT NULL,
 `data`             DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
 PRIMARY KEY(`codice`,`codice_operatore`),
 FOREIGN KEY(`codice_operatore`) REFERENCES `operatore`(`codice_fiscale`)
 ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE `Riceve`
(
 `codice_art`   CHAR(4) NOT NULL,
 `codice_ord_c` CHAR(4) NOT NULL,
 `qnt`          INT(10) UNSIGNED NOT NULL,
 `stato_art`    ENUM('In Coda','In Esecuzione','Finito') DEFAULT NULL,
 PRIMARY KEY (`codice_art`,`codice_ord_c`),
 FOREIGN KEY (`codice_art`) REFERENCES `articolo` (`codice`) ON DELETE
 CASCADE ON UPDATE CASCADE,
 FOREIGN KEY(`codice_ord_c`) REFERENCES `ordine_c` (`codice`) ON DELETE
 CASCADE ON UPDATE CASCADE
);

CREATE TABLE `Richiede`
(
 `codice_for`   CHAR(16) NOT NULL,
 `codice_ord_o` CHAR(4) NOT NULL,
 `qnt`          INT(10) UNSIGNED NOT NULL,
 PRIMARY KEY (`codice_for`,`codice_ord_o`),
 FOREIGN KEY (`codice_for`) REFERENCES `fornitore` (`codice_fiscale`) ON
 DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (`codice_ord_o`) REFERENCES `ordine_o` (`codice`) ON DELETE
```

```
 CASCADE ON UPDATE CASCADE
);

CREATE TABLE `Utensile`
(
 `codice`      CHAR(4) NOT NULL,
 `codice_ccl` CHAR(4) DEFAULT NULL,
 `nome`        VARCHAR(25) NOT NULL,
 `tempo_prp`   TIME NOT NULL DEFAULT '00:05:00',
 PRIMARY KEY (`codice`),
 FOREIGN KEY (`codice_ccl`) REFERENCES `ciclo` (`codice`) ON DELETE SET NULL
 ON UPDATE CASCADE
);

Elimina Macchina
CREATE FUNCTION `elimina_macchina`(`codice`char(4))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGINIF NOT EXISTS
(
        SELECT *
        FROM macchine
        WHERE macchine.codice = codice
)
then
        RETURN 'Errore codice macchina non trovato';
else
        DELETE
FROM macchine
        WHERE macchine.codice = codice;RETURN 'Eseguito';ENDIF;END ;;DELIMITER ;
DELIMITER ;;


Elimina articolo
CREATE FUNCTION `elimina_articolo`(`codice` char(4))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGIN
 IF NOT EXISTS
(
SELECT*
FROM articolo
WHERE articolo.codice=codice)THEN
 RETURN 'codice Articolo non presente';
 else
  DELETE
  FROM articolo
  WHERE articolo.codice = codice;
return 'Eseguito';
endIF;END


Modifica stato ordine cliente
CREATE FUNCTION `modifica_stato_ordine_cliente`(`codice` char(4),
`stato` enum('Accettato',
'In Esecuzione',
'Terminato'))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGIN
  IF NOT EXISTS
  (
SELECT*
FROM ordine_c
WHERE ordine_c.codice = codice) THEN
```

```
RETURN'Codice non esistente';
elseif ((
(
        SELECTordine_c.stato
        FROM ordine_c
        WHERE ordine_c.codice=codice)='NonAccettato')
        AND ( stato = 1) THEN
UPDATE ordine_c SET ordine_c.stato = stato
WHERE ordine_c.codice = codice;
return'Eseguito';
elseif((
(
SELECT ordine_c.stato
FROM ordine_c
WHERE ordine_c.codice = codice) = 'Accettato')
AND
stato = 2) THEN
 UPDATE ordine_c SET ordine_c.stato = stato
 WHERE ordine_c.codice = codice
 return 'Eseguito';
elseif ((
(
SELECT ordine_c.stato
FROM ordine_c
WHERE ordine_c.codice = codice) = 'In Esecuzione')
AND
stato = 3) THEN
UPDATEordine_c
SETordine_c.stato = stato
WHERE ordine_c.codice = codice;
return'Eseguito';
else
RETURN 'Non Eseguito';
end
IF;END;;DELIMITER;


Modifica tempo preparazione utensile
DELIMITER;;
CREATE FUNCTION `modifica_tmp_prep_utensile`(`codice`char(4),
`tempo_prp`varchar(25))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGIN
IF NOT EXISTS
(
SELECT*
FROM utensile
WHEREutensile.codice = codice)THEN
RETURN'Codice non esistente';
else
UPDATE utensile
SET utensile.tempo_prp = time(tempo_prp)
WHERE utensile.codice =codice;
return 'Eseguito';
endIF;END;;
DELIMITER;
```

```
Aggiungi materia prima
CREATE FUNCTION `aggiungi_materia_prima`(`codice`char(4),
`descrizione`varchar(255),
`materiale`varchar(25),
`categoria`varchar(25),
`stato`varchar(25),
`unit`enum('kg',
'm',
'lt'),
`qnt`int(10)unsigned,
`codice_fiscale`char(16),
`nome`varchar(25),
`cognome`varchar(25),
`descrizione_f`varchar(255),
`prezzo`int(10),
`tempo_cng`int(10),
`soglia`int unsigned)
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGIN
IF EXISTS
(
SELECT*
FROM fornitore
WHERE fornitore.codice_fiscale = codice_fiscale) THEN
RETURN'Fornitore esistente';
elseif EXISTS
(
SELECT*
FROM materie_prime
WHERE materie_prime.codice = codice)THEN
RETURN'Codice Materia Prima già esistente';
else
INSERT INTO materie_prime VALUES
(
codice,
descrizione,
materiale,
categoria,
stato,
unit,
qnt,
soglia
);
insert INTO fornitore VALUES
(
codice_fiscale,
codice,
nome,
cognome,
descrizione_f,
prezzo,
time(tempo_cng)
);
return 'Eseguito';
endIF;END;;
```

```
Aggiungi articolo ad ordine cliente
CREATE FUNCTION `aggiungi_articolo_ad_ordine_c`(`codice_ordine`char(4),
`codice_articolo`char(4),
`qnt`int(10))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGIN
IF NOT EXISTS
(
SELECT*
FROM ordine_c
WHERE ordine_c.codice = codice_ordine
)
THEN
RETURN 'Codice Ordine non esistente';
elseif NOT EXISTS
(
SELECT*
FROM articolo
WHERE articolo.codice = codice_articolo
)
THEN
RETURN'Codice Articolo non esistente';
elseif EXISTS
(
SELECT*
FROMriceve
WHEREriceve.codice_art=codice_articolo
ANDriceve.codice_ord_c=codice_ordine
)
THEN
RETURN'Associazione già esistente';
else
INSERT INTO riceve VALUES
(
codice_articolo,
codice_ordine,
qnt,
NULL
);
return 'Eseguito';
endIF;END;; DELIMITER ;


Visualizza materia prima
delimiter;;
CREATE PROCEDURE`visualizza_materia_prima`(IN `codice` char(4),
IN`mat`char(25))nosql
BEGIN
IF(mat!=''
AND
codice!='')THEN
SELECT*
FROMmaterie_prime
WHEREmaterie_prime.codice=codice
AND materie_prime.materiale=mat;
endIF;IF(mat=''
AND
codice!='')then
SELECT*
FROMmaterie_prime
WHEREmaterie_prime.codice=codice;ENDIF;IF(mat!=''
AND
codice='')then
```

```
SELECT*
FROMmaterie_prime
WHEREmaterie_prime.materiale=mat;ENDIF;IF(mat=''
AND
codice='')then
SELECT*
FROMmaterie_prime;
ENDIF;END;;DELIMITER;


Aggiungi ordine cliente
CREATE FUNCTION `aggiungi_ordine_cliente`(`codice`char(4),
`codice_clt char(16),
`data_creaz`varchar(25),
`descrizione`varchar(255),
`codice_art`char(4),
`qnt`int(10))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci no sql
BEGIN
IF EXISTS
(
SELECT*
FROM ordine_c
WHERE ordine_c.codice = codice
)
THEN
RETURN 'Codice Ordine già esistente';
elseif NOT EXISTS
(
SELECT*
FROMcliente
WHEREcliente.codice_fiscale=codice_clt
)
THEN
RETURN 'Cliente non esistente';
elseif NOT EXISTS
(
SELECT*
FROMarticolo
WHEREarticolo.codice=codice_art
)
THEN
RETURN'Articolo non esistente';
else
INSERT INTO ordine_c VALUES
(
codice,
codice_clt,
0,
time(data_creazione),
1,
descrizione
);
insert INTO riceve VALUES
(
codice_art,
codice,
qnt,
NULL
);
return'Eseguito';
endIF;END;;DELIMITER;
```

```
Aggiungi Manutenzione interna
CREATEFUNCTION`aggiungi_manutenzione_interna`(`codice`char(4),
`codice_mac`char(4),
`codice_operaio`char(16),
`data`varchar(25),
`descrizione`varchar(255))
returns varchar(255) charset utf8 COLLATE utf8_unicode_ci
BEGIN
 IF EXISTS
 (
SELECT*
FROM manutenzione
manutenzione.codice=codice)THEN
RETURN 'Codice già esistente';
elseif NOT EXISTS
(
SELECT*
FROM macchine
WHERE macchine.codice=codice_mac)THEN
RETURN 'macchina non esistente';
elseif NOT EXISTS
(
SELECT*
FROM operaio
WHERE operaio.codice_fiscale= codice_operaio) THEN
 RETURN'operaio non esistente';
else
INSERT INTO manutenzione VALUES
(
codice,
codice_mac,
time(data),
descrizione
);

insert INTO interna VALUES
(
codice,
codice_operaio
);
return 'Eseguito';
endIF;END ;;DELIMITER ;
ELIMINA delimiter;;




Aggiungi articolo
function`aggiungi_articolo`(`codice_articolo`CHAR(4),
`descrizione`VARCHAR(255),
`costo_prd`INT(10),
`prezzo_vnd`INT(10),
`codice_ciclo`CHAR(4),
`tempo_plz`VARCHAR(25),
`codice_mat`CHAR(4),
`qnt_mat`INT(10),
`codice_utensile`CHAR(4),
`nome_utensile`VARCHAR(25),
`tempo_prp_utensile`VARCHAR(25),
`codice_operazione`CHAR(4))returnsVARCHAR(255)charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
```

```
IF EXISTS
(
SELECT*
WHERE articolo.codice = codice_articolo) THEN
RETURN 'Codice articolo esistente';ELSEIF
EXISTS
(
SELECT*
FROM ciclo
WHERE ciclo.codice = codice_ciclo) THEN
RETURN 'Codice ciclo esistente';ELSEIF
NOT EXISTS
(
SELECT*
FROM materie_prime
WHEREmaterie_prime.codice = codice_mat)THEN
RETURN 'Codice materia prima non esistente';ELSEIF
EXISTS
(
SELECT*
FROM utensile
WHERE utensile.codice = codice_utensile) THEN
RETURN 'Codice utensile già esistente';ELSEIF
EXISTS
(
SELECT*
FROM fase
WHERE fase.codice_ccl = codice_ciclo
AND fase.codice_operazioni= codice_operazione) THEN
RETURN 'Operazione già esistente nel ciclo';ELSE
INSERT INTO articolo VALUES
(
codice_articolo,
descrizione,
costo_prd,
prezzo_vnd,
time('00:00:00')
);INSERT INTO ciclo VALUES
( codice_ciclo, codice_articolo, descrizione, costo_prd, prezzo_vnd,
tim
Time('00:00:00'),
Time(tempo_plz)
);INSERT INTO necessita VALUES
(
codice_mat,
codice_articolo,
qnt_mat );INSERT INTOutensile VALUES
(
codice_utensile,
codice_ciclo,
nome_utensile,
Time(tempo_prp_utensile)
);INSERT INTO fase VALUES (
1,
codice_operazione,
codice_ciclo
);RETURN 'Eseguito';ENDIF;END
```

```
Aggiungi cliente
function `aggiungi_cliente`(`codice_fiscale` CHAR(16),
`nome`VARCHAR(25),
`cognome`VARCHAR(25))returnsVARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
begin
IF EXISTS
(
SELECT*
FROM cliente
WHERE cliente.codice_fiscale = codice_fiscale) THEN
RETURN'Cliente esistente';ELSE
INSERT INTO clienteVALUES
( codice_fiscale,nome, cognome);RETURN'Eseguito';ENDIF;END

Aggiungi fornitore
function`aggiungi_fornitore`(`codice_fiscale`CHAR(16),
`codice_mat`CHAR(4),
`nome`VARCHAR(25),
`cognome`VARCHAR(25),
`descrizione`VARCHAR(255),
`prezzo`INT(10)UNSIGNED,
`tempo_cng`INT(10)UNSIGNED)returns VARCHAR(255)charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
IF EXISTS
(
SELECT*
FROM fornitore
WHERE fornitore.codice_fiscale = codice_fiscale)THEN
RETURN'Fornitore esistente';ELSEIF
NOT EXISTS
(
SELECT*
FROM materie_prime
WHERE materie_prime.codice=codice_mat)THEN
RETURN'Materia non esistente';ELSE
INSERT INTO fornitoreVALUES
(
codice_fiscale,
codice_mat,
nome,
cognome,
descrizione,
prezzo,
tempo_cng
);RETURN 'Eseguito';END IF;END



Aggiungi macchina
 function `aggiungi_macchina`(`codice` CHAR(4),
`descrizione`VARCHAR(255),
`in_uso`enum('Si', 'No'),
`codice_op`CHAR(4),
`durata`time)returnsVARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
IF NOT EXISTS
(
SELECT*
FROM operazioni
WHERE operazioni.codice = codice_op) THEN
RETURN'Operazione non esistente';ELSEIF
```

```
NOT EXISTS
(
SELECT*
FROM articolo,
ciclo,
fase,
operazioni,
esegue
WHEREarticolo.codice=ciclo.codice_art
ANDciclo.codice=fase.codice_ccl
ANDfase.codice_operazioni=operazioni.codice
ANDoperazioni.codice=codice_op) THEN
RETURN'Nessun articolo esegue questa operazione';ELSEIF
EXISTS
(
SELECT*
FROM esegue
WHERE  esegue.codice_operazioni = codice_op) THEN
RETURN'Operazione gia associata';ELSEIF
EXISTS
(
SELECT*
FROMmacchine
WHEREmacchine.codice=codice)THEN
RETURN'Codice macchina gia esistente';ELSE
INSERT INTO macchine VALUES
( codice,
descrizione,
in_uso
);INSERT INT Oesegue VALUES
( codice_op, codice, Time(durata) );RETURN'Eseguito';END IF;END

Aggiungi manutenzione esterna
function `aggiungi_manutenzione_esterna`(`codice` CHAR(4),
`codice_mac`CHAR(4),
`data_m`VARCHAR(25),
`descrizione`VARCHAR(255),
`pezzo_ricambio`CHAR(30),
`ditta`CHAR(25))returns VARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
begin
IF EXISTS
(
SELECT*
FROM manutenzione
WHERE manutenzione.codice = codice)
THEN
RETURN 'Codice già esistente';ELSEIF
NOT EXISTS
(
SELECT*
FROM macchine
WHERE macchine.codice = codice_mac)
THEN
RETURN 'Codice macchina non esistente';ELSE
INSERT INTO manutenzione VALUES
(
codice,
codice_mac,
time(data_m),
descrizione );INSERT INTO esterna VALUES (
codice,
pezzo_ricambio,
```

```
ditta
);RETURN 'Eseguito';END IF;END
```

```
Aggiungi materia ad ordine operatore
function `aggiungi_materia_ad_ordine_o`(`codice_ordine` CHAR(4),
`codice_fornitore` CHAR(16),
`qnt`INT(10)) returns VARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
IF NOT EXISTS
(
SELECT*
FROM ordine_o
WHERE ordine_o.codice = codice_ordine) THEN
RETURN 'Codice Ordine non esistente';ELSEIF
NOT EXISTS
(
SELECT*
FROM fornitore
WHERE fornitore.codice_fiscale = codice_fornitore) THEN
RETURN 'Codice Fornitore non esistente';ELSEIF
EXISTS
(
SELECT*
FROM richiede
WHERE  richiede.codice_for = codice_fornitore
             AND    richiede.codice_ord_o = codice_ordine) THEN
      RETURN 'Associazione già esistente';ELSE
      INSERT INTO richiede VALUES
       (
            codice_fornitore,
            codice_ordine,
            qnt
       );RETURN 'Eseguito';END IF;END
```

```
Aggiungi materia prima
function `aggiungi_materia_prima`(`codice`CHAR(4),
                           `descrizione`VARCHAR(255),
                           `materiale`VARCHAR(25),
                           `categoria`VARCHAR(25),
                           `stato`VARCHAR(25),
                           `unit`enum('kg',
                           'm',
                           'lt'),
                           `qnt`INT(10)UNSIGNED,
                           `codice_fiscale`CHAR(16),
                           `nome`VARCHAR(25),
                           `cognome`VARCHAR(25),
                           `descrizione_f`VARCHAR(255),
                           `prezzo`INT(10),
                           `tempo_cng`INT(10),
```

```
                            `soglia`INT UNSIGNED) returns VARCHAR(255) charset utf8
COLLATE utf8_unicode_ci noSQL
begin
IF EXISTS
(
SELECT*
FROMfornitore
WHERE fornitore.codice_fiscale = codice_fiscale) THEN
RETURN'Fornitore esistente';ELSEIF
EXISTS
(
SELECT*
FROM materie_prime
WHERE materie_prime.codice = codice)THEN
RETURN'Codice Materia Prima già esistente';ELSE
INSERT INTO materie_prime VALUES
(
                            codice,
                            descrizione,
                            materiale,
                            categoria,
                            stato,
                            unit,
                            qnt,
                            0,
                            soglia
            );INSER INTO fornitore VALUES
            (
                            codice_fiscale,
                            codice,
                            nome,
                            cognome,
                            descrizione_f,
                            prezzo,
                            Time(tempo_cng)
            );RETURN 'Eseguito';END IF;END

Aggiungi operaio
function`aggiungi_operaio`(`codice_fiscale`CHAR(16),
`nome`VARCHAR(25),
`cognome`VARCHAR(25))returnsVARCHAR(255)charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
IF EXISTS
(
SELECT*
FROMoperaio
WHEREoperaio.codice_fiscale = codice_fiscale) THEN
RETURN'Operaio esistente';ELSE
INSER TINTO operaio VALUES
(
codice_fiscale,
nome,
cognome
);RETURN'Eseguito';ENDIF;END



Aggiungi operatore
function`aggiungi_operatore`(`codice_fiscale`CHAR(16),
`nome` VARCHAR(25),
`cognome`VARCHAR(25)) returns VARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
begin
```

```
IF EXISTS
(
SELECT*
FROMoperatore
WHEREoperatore.codice_fiscale = codice_fiscale) THEN
RETURN'Operatore esistente';ELSE
INSERT INTO operatore VALUES
(
codice_fiscale,
nome,
cognome
);RETURN 'Eseguito';END IF;END

Aggiungi operazione
function`aggiungi_operazione`(`codice`CHAR(4),
`descrizione`VARCHAR(255)) returns VARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
IF EXISTS
(
SELECT*
FROM operazioni
WHERE operazioni.codice = codice) THEN
RETURN 'Codice già esistente';ELSE
INSERT INTO operazioni VALUES
(
codice,
descrizione
);RETURN 'Eseguito';END IF;END



Aggiungi ordine cliente
function `aggiungi_ordine_cliente`(`codice`      CHAR(4),
`codice_clt`  CHAR(16),
`data_creaz`  VARCHAR(25),
`descrizione` VARCHAR(255),
                                   `codice_art`  CHAR(4),
                                   `qnt`         INT(10)) returns VARCHAR(255) char
set utf8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF EXISTS
      (
             SELECT*
             FROM   ordine_c
             WHERE  ordine_c.codice = codice) THEN
      RETURN 'Codice Ordine già esistente';ELSEIF
      NOT EXISTS
      (
             SELECT*
             FROM   cliente
             WHERE  cliente.codice_fiscale = codice_clt)THEN
      RETURN 'Cliente non esistente';ELSEIF
      NOT EXISTS
      (
             SELECT*
             FROM   articolo
             WHERE  articolo.codice = codice_art)THEN
RETURN 'Articolo non esistente';ELSE
INSERT INTOordine_c VALUES
(
codice,
```

```
codice_clt,
0,
time(data_creazione),
1,
descrizione
);INSERT INTO riceve VALUES
(
codice_art,
codice,
qnt,
NULL
);RETURN 'Eseguito';END IF;END


Aggiungi Ordine Operatore
function `aggiungi_ordine_operatore`
              (`codice`           CHAR(4),
              `codice_operatore` CHAR(16),
              `data_o`           VARCHAR(255),
              `codice_for`       CHAR(16),
              `qnt`INT(10))
returns VARCHAR(255) charset utf8 COLLATE utf8_unicode_ci
no SQL
begin
IF EXISTS
        (
              SELECT
              FROM   ordine_o
              WHERE  ordine_o.codice = codice) THEN
        RETURN'Codice Ordine già esistente';ELSEIF
        NOT EXISTS
        (
              SELECT*
              FROM   operatore
              WHERE  operatore.codice_fiscale = codice_operatore)THEN
    RETURN 'Operatore non esistente';ELSEIF
      NOT EXISTS
      (
              SELECT*
              FROM   fornitore
              WHERE  fornitore.codice_fiscale = codice_for)THEN
        RETURN 'Fornitore non esistente';ELSE
        INSERT INTO ordine_o VALUES
                  (
                            codice,
                            codice_operatore,
                            time(data_o)
                  );INSERT INTO richiede VALUES
                  (
                            codice_for,
                            codice,
                            qnt
        );RETURN 'Eseguito';END IF;END


Aggiungi Utensili
  function `aggiungi_utensile`(`codice`     CHAR(4),
                               `codice_ccl` CHAR(4),
                               `nome`       VARCHAR(25),
                               `tempo_prp`  VARCHAR(25)) returns VARCHAR(255) charset
utf8 COLLATE utf8_unicode_ci
    no SQL
  begin
```

```
    IF (codice_ccl = '') THEN
      SET codice_ccl = NULL;END IF;IF ((codice_ccl != NULL) AND NOT EXISTS
      (
             SELECT*
             FROM   ciclo
             WHERE  ciclo.codice = codice_ccl)) THEN
      RETURN 'Ciclo non esistente';ELSEIF
      EXISTS
      (
             SELECT*
             FROM   utensile
             WHERE  utensile.codice = codice) THEN
      RETURN 'Codice già esistente';ELSE
      INSERT INTO utensile VALUES
                  (
                              codice,
                              codice_ccl,
                              nome,
                              time(tempo_prp)
                  );RETURN 'Eseguito';END IF;END


Elimina articolo
  function `elimina_articolo`(`codice` CHAR(4)) returns VARCHAR(255) charset utf8 COL
LATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT*
             FROM   articolo
             WHERE  articolo.codice = codice)
      THEN
      RETURN 'codice Articolo non presente';ELSE
      DELETE
      FROM   articolo
      WHERE  articolo.codice = codice;RETURN 'Eseguito';END IF;END

Elimina Cliente
  function `elimina_cliente`(`codice` CHAR(16)) returns VARCHAR(255) charset utf8 COL
LATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT*
             FROM   cliente
             WHERE  cliente.codice_fiscale = codice)
      THEN
      RETURN 'codice cliente non presente';ELSE
      DELETE
      FROM   cliente
      WHERE  cliente.codice_fiscale = codice;RETURN 'Eseguito';END IF;END


Elimina Macchina
  function `elimina_macchina`(`codice` CHAR(4)) returns VARCHAR(255) charset utf8 COL
LATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT *
```

```
            FROM   macchine
            WHERE  macchine.codice = codice )THEN
      RETURN 'Errore codice macchina non trovato';ELSE
      DELETE
      FROM   macchine
      WHERE  macchine.codice = codice;RETURN 'Eseguito';END IF;END


Elimina manutenzione
  function `elimina_manutenzione`(`codice` CHAR(4)) returns VARCHAR(255) charset utf8
COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
            SELECT *
            FROM   manutenzione
            WHERE  manutenzione.codice = codice )
      THEN
      RETURN'codice Manutenzione non trovato';ELSE
      DELETE
      FROM   manutenzione
      WHERE  manutenzione.codice = codice;RETURN'Eseguito';END IF;END

Elimina materia prima
  function `elimina_materia_prima`(`codice` CHAR(4)) returns VARCHAR(255) charset utf
8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
            SELECT *
            FROM   materie_prime
            WHERE  materie_prime.codice = codice )
      THEN
      RETURN 'Codice Materia Prima non trovata';ELSEIF
      EXISTS
      (
            SELECT *
            FROM   fornitore
            WHERE  codice = fornitore.codice_mat )THEN
      DELETE
      FROM   fornitore
      WHERE  fornitore.codice_mat = codice;DELETE
      FROM   materie_prime
      WHERE  codice = materie_prime.codice;RETURN 'Eseguito, eliminata materia prima
e fornitore associato';ELSE
      DELETE
      FROM   materie_prime
      WHERE  codice = materie_prime.codice;RETURN 'Eseguito, eliminata solo materia p
rima';END IF;END


Elimina operaio
  function `elimina_operaio`(`codice` CHAR(16)) returns VARCHAR(255) charset utf8 CO
LLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
            SELECT *
            FROM   operaio
            WHERE  operaio.codice_fiscale = codice )
```

```
      THEN
      RETURN'codice operaio non trovato';ELSE
      DELETE
      FROM   operaio
      WHERE  operaio.codice_fiscale = codice;RETURN'Eseguito';END IF;END


Elimina operatore
  function `elimina_operatore`(`codice` CHAR(16)) returns VARCHAR(255) charset utf8
COLLATE utf8_unicode_ci
  begin
    IF EXISTS
      (
             SELECT *
             FROM   operatore
             WHERE  operatore.codice_fiscale = codice )THEN
      DELETE
      FROM   operatore
      WHERE  operatore.codice_fiscale = codice;RETURN 'Eseguito';ELSE
      RETURN 'ERROR: nessun codice operatore trovato';END IF;END



 Elimina operazione
  function `elimina_operazione`(`codice` CHAR(4)) returns VARCHAR(255) charset utf8 C
OLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT *
             FROM   operazioni
             WHERE  operazioni.codice = codice )
      THEN
      RETURN 'codice operazione non trovato';ELSE
      DELETE
      FROM   operazioni
      WHERE  operazioni.codice = codice;RETURN 'Eseguito';END IF;END



Elimina ordine cliente
  function `elimina_ordine_cliente`(`codice` CHAR(04)) returns VARCHAR(255) charset u
tf8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT *
             FROM   ordine_c
             WHERE  ordine_c.codice = codice )
      THEN
      RETURN 'Codice ordine non trovato';ELSE
      DELETE
      FROM   ordine_c
      WHERE  codice = ordine_c.codice;RETURN 'Eseguito';END IF;END
```

```
 Elimina ordine operatore
  function `elimina_ordine_operatore`(`codice` CHAR(4)) returns VARCHAR(255) charset
utf8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT *
             FROM   ordine_o
             WHERE  ordine_o.codice = codice )
      THEN
      RETURN 'Codice ordine non trovato';ELSE
      DELETE
      FROM   ordine_o
      WHERE  codice = ordine_o.codice;RETURN 'Eseguito';END IF;END



 Elimina ordine operatore
  function `elimina_ordine_operatore`(`codice` CHAR(4)) returns VARCHAR(255) charset
utf8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT *
             FROM   ordine_o
             WHERE  ordine_o.codice = codice )
      THEN
      RETURN 'Codice ordine non trovato';ELSE
      DELETE
      FROM   ordine_o
      WHERE  codice = ordine_o.codice;RETURN 'Eseguito';END IF;END



 Elimina utensile
  function `elimina_utensile`(`codice` CHAR(4)) returns VARCHAR(255) charset utf8 COL
LATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
             SELECT *
             FROM   utensile
             WHERE  utensile.codice = codice )THEN
      RETURN 'Codice Utensile non presente';ELSE
      DELETE
      FROM   utensile
      WHERE  utensile.codice = codice;RETURN 'Eseguito';END IF;END



  modifica qnt articolo ordine cliente
  function `modifica_qnt_articolo_ordine_c`(`codice_ordine`   INT(4),
                                            `codice_articolo` INT(4),
                                            `qnt`             INT(10)) returns VARCHA
R(255) charset utf8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
```

```
            SELECT*
            FROM   riceve
            WHERE  riceve.codice_ord_c = codice_ordine
            AND    riceve.codice_art = codice_articolo) THEN
      RETURN 'Associazione o codici non esistenti';ELSE
      UPDATE riceve
      SET    riceve.qnt = qnt
      WHERE  riceve.codice_ord_c = codice_ordine
      AND    riceve.codice_art = codice_articolo;RETURN 'Eseguito';END IF;END



Modifica stato ordine cliente
  function `modifica_stato_ordine_cliente`(`codice` CHAR(4),
                                           `stato` enum('Accettato',
                                                        'In Esecuzione',
                                                        'Terminato')) returns VARCHAR
(255) charset utf8 COLLATE utf8_unicode_ci
    no SQL
  begin
    IF NOT EXISTS
      (
            SELECT*
            FROM   ordine_c
            WHERE  ordine_c.codice = codice) THEN
      RETURN 'Codice non esistente';ELSEIF
            ((
               (
               SELECT ordine_c.stato
               FROM   ordine_c
               WHERE  ordine_c.codice = codice) = 'Non Accettato') AND
            (
              stato = 1
            )
            ) THEN
      UPDATE ordine_c SET ordine_c.stato = stato
      WHERE  ordine_c.codice = codice;RETURN 'Eseguito';ELSEIF
            ((
               (
               SELECT ordine_c.stato
               FROM   ordine_c
               WHERE  ordine_c.codice = codice) = 'Accettato') AND stato = 2) THEN
      UPDATE ordine_c SET ordine_c.stato = stato
      WHERE  ordine_c.codice = codice;RETURN 'Eseguito';ELSEIF
            ((
               (
               SELECT ordine_c.stato
               FROM   ordine_c
               WHERE  ordine_c.codice = codice) = 'In Esecuzione') AND stato = 3) THE
N
      UPDATE ordine_c
      SET    ordine_c.stato = stato
      WHERE  ordine_c.codice = codice;RETURN 'Eseguito';ELSE
      RETURN 'Non Eseguito';END IF;END



Visualizza articolo ciclo
PROCEDURE `visualizza_articolo_ciclo`(IN `codice` CHAR(4))
  no SQL
begin
  IF (codice != '') THEN
```

```
    SELECT articolo.*,
           ciclo.codice AS codice_ciclo,
           ciclo.tempo_prp,
           ciclo.tempo_plz
    FROM   articolo,
           ciclo
    WHERE  ciclo.codice_art = articolo.codice
    AND    articolo.codice = codice;END IF;IF (codice = '') THEN
    SELECT articolo.*,
           ciclo.codice AS codice_ciclo,
           ciclo.tempo_prp,
           ciclo.tempo_plz
    FROM   articolo,
           ciclo
    WHERE  ciclo.codice_art = articolo.codice;END IF;END


Visualizza cliente
PROCEDURE `visualizza_cliente`(IN `codice` CHAR(16))
  no SQL
begin
  IF (codice != '') THEN
    SELECT *
    FROM   cliente
    WHERE  cliente.codice_fiscale = codice;END IF;IF (codice = '') THEN
    SELECT *
    FROM   cliente;END IF;END



Visualizza fornitore
PROCEDURE `visualizza_fornitore`(IN `codice`     CHAR(16),
                                 IN `codice_mat` CHAR(4))
  no SQL
begin
  IF (codice != '' AND codice_mat != '') THEN
    SELECT *
    FROM   fornitore
    WHERE  fornitore.codice_fiscale = codice
    AND    fornitore.codice_mat = codice_mat;END IF;IF (codice != '' AND codice_mat =
'') THEN
    SELECT *
    FROM   fornitore
    WHERE  fornitore.codice_fiscale = codice;END IF;IF (codice = '' AND codice_mat !=
'') THEN
    SELECT *
    FROM   fornitore
    WHERE  fornitore.codice_mat = codice_mat;END IF;IF (codice = '' AND codice_mat =
'') THEN
    SELECT *
    FROM   fornitore;END IF;END



Visualizza macchine
PROCEDURE `visualizza_mac`(IN `codice` CHAR(4))
  no SQL
begin
  IF (codice != '') THEN
    SELECT *
    FROM   macchine
```

```
    WHERE  macchine.codice = codice;END IF;IF (codice = '') THEN
    SELECT *
    FROM   macchine;END IF;END


 Visualizza manutenzione
PROCEDURE `visualizza_manutenzione`(IN `codice`     CHAR(4),
                                    IN `codice_mac` CHAR(4),
                                    IN `tipo` enum('Interna','Esterna','Entrambe'),
                                    IN `codice_operaio` CHAR(16))
  no SQL
begin
  IF (codice = '' AND codice_mac = '' AND tipo = 1 AND codice_operaio = '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice;END IF;IF (codice = '' AND codice_mac
= '' AND tipo = 1 AND codice_operaio != '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
    AND    interna.codice_operaio = codice_operaio;END IF;IF (codice = '' AND codice_
mac = '' AND tipo = 2) THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           esterna.pezzo_ricambio,
           esterna.ditta
    FROM   manutenzione,
           esterna
    WHERE  manutenzione.codice = esterna.codice;END IF;IF (codice = '' AND codice_mac
= '' AND tipo = 3) THEN
    SELECT *
    FROM   manutenzione;END IF;IF (codice != '' AND codice_mac = '' AND tipo = 1 AND
codice_operaio = '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
    AND    manutenzione.codice = codice;END IF;IF (codice != '' AND codice_mac = '' A
ND tipo = 1 AND codice_operaio != '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
```

```
    AND    manutenzione.codice = codice
    AND    interna.codice_operaio = codice_operaio;END IF;IF (codice != '' AND codice
_mac = '' AND tipo = 2) THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           esterna.pezzo_ricambio,
           esterna.ditta
    FROM   manutenzione,
           esterna
    WHERE  manutenzione.codice = esterna.codice
    AND    manutenzione.codice = codice;END IF;IF (codice != '' AND codice_mac = '' A
ND tipo = 3) THEN
    SELECT *
    FROM   manutenzione
    WHERE  manutenzione.codice = codice;END IF;IF (codice = '' AND codice_mac != '' A
ND tipo = 1 AND codice_operaio = '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
    AND    manutenzione.codice_mac = codice_mac;END IF;IF (codice = '' AND codice_mac
!= '' AND tipo = 1 AND codice_operaio != '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
    AND    manutenzione.codice_mac = codice_mac
    AND    interna.codice_operaio = codice_operaio;END IF;IF (codice = '' AND codice_
mac != '' AND tipo = 2) THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           esterna.pezzo_ricambio,
           esterna.ditta
    FROM   manutenzione,
           esterna
    WHERE  manutenzione.codice = esterna.codice
    AND    manutenzione.codice_mac = codice_mac;END IF;IF (codice = '' AND codice_mac
!= '' AND tipo = 3) THEN
    SELECT *
    FROM   manutenzione
    WHERE  manutenzione.codice_mac = codice_mac;END IF;IF (codice != '' AND codice_ma
c != '' AND tipo = 1 AND codice_operaio = '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
    AND    manutenzione.codice_mac = codice_mac
```

```
    AND    manutenzione.codice = codice;END IF;IF (codice != '' AND codice_mac != ''
AND tipo = 1 AND codice_operaio != '') THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           interna.codice_operaio
    FROM   manutenzione,
           interna
    WHERE  manutenzione.codice = interna.codice
    AND    manutenzione.codice_mac = codice_mac
    AND    manutenzione.codice = codice
    AND    interna.codice_operaio = codice_operaio;END IF;IF (codice != '' AND codice
_mac != '' AND tipo = 2) THEN
    SELECT manutenzione.codice,
           manutenzione.codice_mac,
           manutenzione.data,
           manutenzione.descrizione,
           esterna.pezzo_ricambio,
           esterna.ditta
    FROM   manutenzione,
           esterna
    WHERE  manutenzione.codice = esterna.codice
    AND    manutenzione.codice_mac = codice_mac
    AND    manutenzione.codice = codice;END IF;IF (codice != '' AND codice_mac != ''
AND tipo = 3) THEN
    SELECT *
    FROM   manutenzione
    WHERE  manutenzione.codice_mac = codice_mac
    AND    manutenzione.codice = codice;END IF;END
```

```
`  Visualizza man macchine
PROCEDURE `visualizza_man_mac`(IN `codice_mac` CHAR(4),
                               IN `codice_man` CHAR(4))
  no SQL
begin
  IF (codice_mac != '' AND codice_man != '') THEN
    SELECT *
    FROM   manutenzione
    WHERE  manutenzione.codice = codice_man
    AND    manutenzione.codice_mac = codice_mac;ELSEIF
    (codice_mac = '' AND codice_man != '')THEN
    SELECT *
    FROM   manutenzione
    WHERE  manutenzione.codice = codice_man;ELSEIF
    (codice_mac != '' AND codice_man = '')THEN
    SELECT macchine.codice          AS 'Codice macchina',
           macchine.descrizione     AS 'Descrizione macchina',
           macchine.in_uso          AS 'In Uso',
           manutenzione.codice      AS 'Codice manutenzione',
           manutenzione.descrizione AS 'Descrizione manutenzione',
           manutenzione.data        AS 'Data manutenzione'
    FROM   manutenzione,
           macchine
    WHERE  macchine.codice = codice_mac
    AND    macchine.codice = manutenzione.codice_mac;ELSEIF
    (codice_mac = '' AND codice_man = '')THEN
    SELECT *
    FROM   manutenzione;END IF;END
```

```
 Visualizza materia articolo
PROCEDURE `visualizza_materia_articolo`(IN `codice_art` VARCHAR(4),
                                        IN `codice_mat` VARCHAR(4))
  no SQL
begin
  IF (codice_mat = '' AND codice_art = '') THEN
    SELECT materie_prime.codice      AS 'codice mat pr',
           materie_prime.descrizione AS 'desc mat',
           materie_prime.materiale   AS 'materiale',
           materie_prime.categoria   AS 'categoria',
           materie_prime.stato       AS 'stato',
           materie_prime.unit?       AS 'unità mat pr',
           materie_prime.soglia,
           materie_prime.qnt,
           materie_prime.qnt_occ,
           necessita.codice_art AS 'Codice art',
           necessita.qnt        AS 'qnt necessaria di art',
           articolo.codice      AS 'codice art',
           articolo.descrizione AS 'desc art',
           articolo.costo_prd   AS 'costo art',
           articolo.prezzo_vnd  AS 'prz vnd art'
    FROM   necessita,
           materie_prime,
           articolo
    WHERE  materie_prime.codice = necessita.codice_mat
    AND    articolo.codice = necessita.codice_art;END IF;IF (codice_mat = '' AND codi
ce_art != '') THEN
    SELECT materie_prime.codice      AS 'codice mat pr',
           materie_prime.descrizione AS 'desc mat',
           materie_prime.materiale   AS 'materiale',
           materie_prime.categoria   AS 'categoria',
           materie_prime.stato       AS 'stato',
           materie_prime.unit?       AS 'unità mat pr',
           materie_prime.soglia,
           materie_prime.qnt,
           materie_prime.qnt_occ,
           necessita.codice_art AS 'Codice art',
           necessita.qnt        AS 'qnt necessaria di art',
           articolo.codice      AS 'codice art',
           articolo.descrizione AS 'desc art',
           articolo.costo_prd   AS 'costo art',
           articolo.prezzo_vnd  AS 'prz vnd art'
    FROM   necessita,
           materie_prime,
           articolo
    WHERE  materie_prime.codice = necessita.codice_mat
    AND    articolo.codice = necessita.codice_art
    AND    articolo.codice = codice_art;END IF;IF (codice_mat != '' AND codice_art =
'') THEN
    SELECT materie_prime.codice      AS 'codice mat pr',
           materie_prime.descrizione AS 'desc mat',
           materie_prime.materiale   AS 'materiale',
           materie_prime.categoria   AS 'categoria',
           materie_prime.stato       AS 'stato',
           materie_prime.unit?       AS 'unità mat pr',
           materie_prime.soglia,
           materie_prime.qnt,
           materie_prime.qnt_occ,
           necessita.codice_art AS 'Codice art',
           necessita.qnt        AS 'qnt necessaria di art',
```

```
            articolo.codice      AS 'codice art',
            articolo.descrizione AS 'desc art',
            articolo.costo_prd   AS 'costo art',
            articolo.prezzo_vnd  AS 'prz vnd art'
    FROM    necessita,
            materie_prime,
            articolo
    WHERE   materie_prime.codice = necessita.codice_mat
    AND     articolo.codice = necessita.codice_art
    AND     materie_prime.codice = codice_mat;END IF;IF (codice_mat != '' AND codice_a
rt != '') THEN
    SELECT materie_prime.codice      AS 'codice mat pr',
            materie_prime.descrizione AS 'desc mat',
            materie_prime.materiale   AS 'materiale',
            materie_prime.categoria   AS 'categoria',
            materie_prime.stato       AS 'stato',
            materie_prime.unit?       AS 'unità mat pr',
            materie_prime.soglia,
            materie_prime.qnt,
            materie_prime.qnt_occ,
            necessita.codice_art AS 'Codice art',
            necessita.qnt        AS 'qnt necessaria di art',
            articolo.codice      AS 'codice art',
            articolo.descrizione AS 'desc art',
            articolo.costo_prd   AS 'costo art',
            articolo.prezzo_vnd  AS 'prz vnd art'
    FROM    necessita,
            materie_prime,
            articolo
    WHERE   materie_prime.codice = necessita.codice_mat
    AND     articolo.codice = necessita.codice_art
    AND     articolo.codice = codice_art
    AND     materie_prime.codice = codice_mat;END IF;END
```

```
 Visualizza operaio
PROCEDURE `visualizza_operaio`(IN `codice` CHAR(16))
  no SQL
begin
  IF (codice = '') THEN
    SELECT *
    FROM   operaio;ELSE
    SELECT *
    FROM   operaio
    WHERE  operaio.codice_fiscale = codice;END IF;END
```

```
 Visualizza operazione macchina
PROCEDURE `visualizza_op_mac`(IN `codice_operazione` CHAR(4),
                              IN `codice_mac`        CHAR(4))
  no SQL
begin
  IF (codice_mac != '' AND codice_operazione != '') THEN
    SELECT macchine.codice        AS 'Codice macchina',
           macchine.descrizione   AS 'Desc macchina',
```

```
        macchine.in_uso        AS 'In uso',
        operazioni.codice      AS 'Codice operazione',
        operazioni.descrizione AS 'Desc op',
        esegue.durata          AS 'Durata Op'
FROM    macchine,
        operazioni,
        esegue
WHERE   macchine.codice = codice_mac
AND     macchine.codice = esegue.codice_mac
AND     esegue.codice_operazioni = operazioni.codice
AND     operazioni.codice = codice_operazione;ELSEIF
(codice_mac = '' AND codice_operazione != '')THEN
SELECT macchine.codice         AS 'Codice macchina',
        macchine.descrizione   AS 'Desc macchina',
        macchine.in_uso        AS 'In uso',
        operazioni.codice      AS 'Codice operazione',
        operazioni.descrizione AS 'Desc op',
        esegue.durata          AS 'Durata Op'
FROM    macchine,
        operazioni,
        esegue
WHERE   macchine.codice = esegue.codice_mac
AND     esegue.codice_operazioni = operazioni.codice
AND     operazioni.codice = codice_operazione;ELSEIF
(codice_mac != '' AND codice_operazione = '')THEN
SELECT macchine.codice         AS 'Codice macchina',
        macchine.descrizione   AS 'Desc macchina',
        macchine.in_uso        AS 'In uso',
        operazioni.codice      AS 'Codice operazione',
        operazioni.descrizione AS 'Desc op',
        esegue.durata          AS 'Durata Op'
FROM    macchine,
        operazioni,
        esegue
WHERE   macchine.codice = codice_mac
AND     macchine.codice = esegue.codice_mac
AND     esegue.codice_operazioni = operazioni.codice;ELSEIF
(codice_mac = '' AND codice_operazione = '')THEN
SELECT macchine.codice         AS 'Codice macchina',
        macchine.descrizione   AS 'Desc macchina',
        macchine.in_uso        AS 'In uso',
        operazioni.codice      AS 'Codice operazione',
        operazioni.descrizione AS 'Desc op',
        esegue.durata          AS 'Durata Op'
FROM    macchine,
        operazioni,
        esegue
WHERE   macchine.codice = esegue.codice_mac
AND     esegue.codice_operazioni = operazioni.codice;END IF;END



  Visualizza ordine cliente
PROCEDURE `visualizza_ordine_c`(IN `codice_ordine`  CHAR(4),
                                IN `codice_cliente` CHAR(16))
  no SQL
  DETERMINISTIC
begin
  IF (codice_cliente != '' AND codice_ordine != '') THEN
    SELECT *
    FROM   ordine_c
```

```
    WHERE  ordine_c.codice = codice_ordine
    AND    ordine_c.codice_clt = codice_cliente;END IF;IF (codice_ordine != '' AND co
dice_cliente = '') THEN
    SELECT *
    FROM   ordine_c
    WHERE  ordine_c.codice = codice_ordine;END IF;IF (codice_ordine = ''AND codice_cl
iente != '') THEN
    SELECT *
    FROM   ordine_c
    WHERE  ordine_c.codice_clt = codice_cliente;END IF;IF (codice_ordine = '' AND cod
ice_cliente = '') THEN
    SELECT ordine_c.codice         AS 'codice ordine',
           ordine_c.codice_clt     AS 'codice cliente',
           ordine_c.data_creazione AS 'data ordine',
           ordine_c.stato          AS 'stato ordine',
           articolo.codice         AS 'codice articolo',
           articolo.descrizione    AS 'descrizione articolo'
    FROM   ordine_c,
           riceve,
           articolo
    WHERE  ordine_c.codice = riceve.codice_ord_c
    AND    riceve.codice_art = articolo.codice;END IF;END
```

```
   Visualizza ordine operatore
PROCEDURE `visualizza_ordine_o`(IN `codice_ordine`    CHAR(4),
                                IN `codice_operatore` CHAR(16))
  no SQL
begin
  IF (codice_ordine != '' AND codice_operatore != '') THEN
    SELECT *
    FROM   ordine_o
    WHERE  ordine_o.codice_operatore = codice_operatore
    AND    ordine_o.codice = codice_ordine;END IF;IF (codice_ordine = '' AND codice_o
peratore != '') THEN
    SELECT *
    FROM   ordine_o
    WHERE  ordine_o.codice_operatore = codice_operatore;END IF;IF (codice_ordine != '
' AND codice_operatore = '') THEN
    SELECT *
    FROM   ordine_o
    WHERE  ordine_o.codice = codice_ordine;END IF;IF (codice_ordine = '' AND codice_o
peratore = '') THEN
    SELECT *
    FROM   ordine_o;END IF;END
```

```
   Visualizza ordine cliente
PROCEDURE `visualizza_ordini_cliente`(IN `codice_cliente` CHAR(16))
  no SQL
begin
  IF EXISTS
    (
           SELECT *
           FROM   cliente,
                  ordine_c
           WHERE  cliente.codice_fiscale = codice_cliente
           AND    cliente.codice_fiscale = ordine_c.codice_clt )THEN
    SELECT cliente.codice_fiscale AS 'cod fisc',
           cliente.nome           AS 'Nome',
```

```
            cliente.cognome         AS 'Cognome',
            ordine_c.codice         AS 'Codice ordine',
            ordine_c.qnt_art        AS 'qnt tot',
            ordine_c.stato          AS 'Stato ordine',
            riceve.stato_art        AS 'Stato art',
            riceve.qnt              AS 'Qnt art',
            articolo.codice         AS 'Codice articolo',
            articolo.descrizione    AS 'Descrizione articolo',
            articolo.costo_prd      AS 'Costo produzione',
            articolo.prezzo_vnd     AS 'Prezzo vendita',
            articolo.tempo_prd      AS 'Tempo preparazione'
    FROM    articolo,
            riceve,
            ordine_c,
            cliente
    WHERE   cliente.codice_fiscale = ordine_c.codice_clt
    AND     cliente.codice_fiscale = codice_cliente
    AND     ordine_c.codice = riceve.codice_ord_c
    AND     riceve.codice_art = articolo.codice;END IF;END



   Visualizza utensile
PROCEDURE `visualizza_utensile`(IN `codice`     CHAR(4),
                                IN `codice_ccl` CHAR(4))
  no SQL
begin
  IF (codice != '' AND codice_ccl != '') THEN
    SELECT *
    FROM   utensile
    WHERE  utensile.codice = codice
    AND    utensile.codice_ccl = codice_ccl;END IF;IF (codice != '' AND codice_ccl =
'') THEN
    SELECT *
    FROM   utensile
    WHERE  utensile.codice = codice;END IF;IF (codice = '' AND codice_ccl != '') THEN

    SELECT *
    FROM   utensile
    WHERE  utensile.codice_ccl = codice_ccl;END IF;IF (codice = '' AND codice_ccl = '
') THEN
    SELECT *
    FROM   utensile;END IF;END
```

Op a) PL/SQL

Vincoli non gestibili in fase di definizione delle tabelle:

- **Soglia minima**: se una materia prima scende sotto la soglia fissata allora l'operatore emette un'ordine del materiale richiedendolo ad uno dei fornitori.
- **Quantità occupata**: se un ordine passa dallo stato “non accettato” allo stato “accettato”, deve essere modificata la quantità occupata e libera delle relative materie prime.
- **Quantità libera:** Se un operatore effettua un ordine si deve aggiornare la quantità disponibile di materie prime.
- **Tempo preparazione utensili**: ogni tempo di preparazione di un ciclo deve essere la somma dei tempi di preparazione utensile.
- **Tempo totale articolo**: ogni tempo totale di un articolo deve essere la somma dei tempi di preparazione e pulizia del ciclo associato e delle somma della durata di ogni operazione.

LISTA DEI TRIGGER CREATI:

- **Before_Insert_Materie_Prime**: Nega la possibilità di impostare il valore iniziale di quantità occupata

```
CREATE TRIGGER `before_insert_materie_prime`
BEFORE INSERT ON `Materie_Prime`
FOR EACH ROW
begin
  SET NEW.qnt_occ = 0;
end
```

- **After_Insert_Richiede**: Aggiunge quantità disponibile a una materia prima.

```
CREATE TRIGGER `after_update_richiede`
AFTER UPDATE ON `Richiede`
  FOR EACH ROW
  begin
      IF NEW.qnt > OLD.qnt THEN
        UPDATE Materie_Prime,Fornitore,Richiede
        SET   Materie_Prime.qnt = Materie_Prime.qnt + ( NEW.qnt - OLD.qnt )
        WHERE Richiede.codice_for = OLD.codice_for
          AND Richiede.codice_for = Fornitore.codice_fiscale
          AND Fornitore.codice_mat = Materie_prime.codice;
      ELSEIF NEW.qnt < OLD.qnt THEN
        UPDATE Materie_Prime,fornitore,richiede
        SET    Materie_Prime.qnt = Materie_Prime.qnt - ( OLD.qnt - NEW.qnt )
        WHERE  Richiede.codice_for = OLD.codice_for
          AND Richiede.codice_for = Fornitore.codice_fiscale
          AND Fornitore.codice_mat = Materie_prime.codice;
      end IF;
  end
```

- **After_Update_Richiede**: Aggiorna le quantità disponibili di una materia prima.

```
CREATE TRIGGER `after_update_ciclo`
AFTER UPDATE ON `Ciclo`
  FOR EACH ROW
  begin
      IF ( NEW.tempo_prp != OLD.tempo_prp ) THEN
        UPDATE Articolo
        SET    Articolo.tempo_prd = Time (tempo_prd + ( NEW.tempo_prp -
                                          OLD.tempo_prp ))
        WHERE  NEW.codice_art = Articolo.codice;
      end IF;
      IF ( NEW.tempo_plz != OLD.tempo_plz ) THEN
        UPDATE Articolo
        SET    Articolo.tempo_prd = Time (tempo_prd + ( NEW.tempo_plz -
                                          OLD.tempo_plz ))
        WHERE NEW.codice_art = Articolo.codice;
      end IF;
  end
```

- **After_Delete_Richiede**: Sottrae la quantità che rendeva disponibile l'ordine ad una materia prima. Questo comporta che tutti gli Ordini_O devono restare in memoria ma verificare l'integrità del dato derivato.

```
CREATE TRIGGER `after_delete_richiede`
AFTER DELETE ON `Richiede`
FOR EACH ROW
begin
  UPDATE Materie_Prime,Fornitore,Richiede
  SET Materie_Prime.qnt = Materie_Prime.qnt - OLD.qnt
  WHERE Richiede.codice_for = OLD.codice_for
         AND Richiede.codice_for = Fornitore.codice_fiscale
         AND Fornitore.codice_mat = Materie_Prime.codice;
end
```

- **Before_Insert_Riceve**: Gestisce la quantità occupata delle materie prime riferendosi allo stato dell'Ordine_C:
  Se stato ordine è “non accettato” vengono aggiornati i valori della quantità in riceve e della quantità in ordine e lo stato dell'articolo è posto uguale a NULL.
  Se stato ordine è “accettato” imposta lo stato dell'articolo uguale a “In coda”.
  Se stato ordine è “in esecuzione” o “terminato” allora l'operazione non è permessa.

```
CREATE TRIGGER `before_insert_riceve`
BEFORE INSERT ON `Riceve`
FOR EACH ROW
  begin
      IF EXISTS (SELECT*
                 FROM Ordine_C
                 WHERE NEW.codice_ord_c = Ordine_C.codice
                   AND Ordine_C.stato = "accettato") THEN
        SET NEW.stato_art = "in coda";
        UPDATE Ordine_C
        SET Ordine_C.qnt_art = Ordine_C.qnt_art + NEW.qnt
        WHERE NEW.codice_ord_c = Ordine_C.codice;

      ELSEIF EXISTS(SELECT *
                    FROM Ordine_C
                    WHERE NEW.codice_ord_c = Ordine_C.codice
                      AND Ordine_C.stato="non accettato" ) THEN
        SET NEW.stato_art = NULL;
        UPDATE Ordine_C
```

```
            SET  Ordine_C.qnt_art = Ordine_C.qnt_art + NEW.qnt
            WHERE NEW.codice_ord_c = Ordine_C.codice;

          ELSEIF EXISTS( SELECT *
                         FROM Ordine_C
                         WHERE NEW.codice_ord_c = Ordine_C.codice
                            AND Ordine_C.stato="in esecuzione" ) THEN
            signal sqlstate '45000'SET message_text = 'Ordine in esecuzione!';
          ELSEIF EXISTS(SELECT *
                        FROM Ordine_C
                        WHERE NEW.codice_ord_c = Ordine_C.codice
                           AND  Ordine_C.stato="terminato" ) THEN
            signal sqlstate '45000' SET message_text = 'Ordine terminato!';
          end IF;
      end
```

- **Before_Update_Riceve**: Gestisce la quantità degli articoli in un ordine al variare del numero della quantità dello specifico articolo ordinato .
  Se stato ordine è “non accettato” o “accettato” viene aggiornata la quantità dell'ordine.
  Se stato ordine è “in esecuzione” o “terminato”non è possibile modificarlo, e restituisce un errore.
  Inoltre quando viene modificato lo stato dell'articolo da “accettato” ad “In esecuzione”, il trigger modifica la quantità e la quantità occupata delle materie prime.

```
CREATE TRIGGER `before_update_riceve` BEFORE
  UPDATE
  ON `riceve` FOR EACH row begin IF ((new.stato_art != 'finito')
  AND    EXISTS
         (
                SELECT *
                FROM   ordine_c
                WHERE  new.codice_ord_c = ordine_c.codice
                AND    ordine_c.stato="in esecuzione" )) THEN signal SQLSTATE '45000'

  SET    message_text = 'Ordine in esecuzione!';

ELSEIF
  EXISTS
  (
         SELECT *
         FROM   ordine_c
         WHERE  new.codice_ord_c = ordine_c.codice
         AND    ordine_c.stato="terminato" ) THEN
  signal SQLSTATE '45000' SET message_text = 'Ordine terminato!';
ELSEIF
  EXISTS
  (
         SELECT *
         FROM   ordine_c
         WHERE  new.codice_ord_c = ordine_c.codice
         AND    ordine_c.stato="accettato" ) THEN
  IF (new.qnt > old.qnt) THEN
    UPDATE ordine_c
    SET    ordine_c.qnt_art = ordine_c.qnt_art + (new.qnt-old.qnt)
    WHERE  new.codice_ord_c = ordine_c.codice;

  end IF;
  IF(new.qnt < old.qnt) THEN
    UPDATE ordine_c
```

```
    SET    ordine_c.qnt_art = ordine_c.qnt_art - (old.qnt-new.qnt)
    WHERE  new.codice_ord_c = ordine_c.codice;

  end IF;
ELSEIF
  EXISTS
  (
         SELECT *
         FROM   ordine_c
         WHERE  new.codice_ord_c = ordine_c.codice
         AND    ordine_c.stato="non accettato" ) THEN
  IF (new.qnt > old.qnt) THEN
    UPDATE ordine_c
    SET    ordine_c.qnt_art = ordine_c.qnt_art + (new.qnt-old.qnt)
    WHERE  new.codice_ord_c = ordine_c.codice;

  end IF;
  IF(new.qnt < old.qnt) THEN
    UPDATE ordine_c
    SET    ordine_c.qnt_art = ordine_c.qnt_art - (old.qnt-new.qnt)
    WHERE  new.codice_ord_c = ordine_c.codice;

  end IF;
ELSEIF
  new.stato_art = "in esecuzione" THEN
  UPDATE materie_prime,
         necessita,
         articolo
  SET    materie_prime.qnt_occ = materie_prime.qnt_occ + ((old.qnt) * necessita.qnt),

         materie_prime.qnt = materie_prime.qnt         - ((old.qnt) * necessita.qnt)
  WHERE  articolo.codice = old.codice_art
  AND    necessita.codice_mat = materie_prime.codice
  AND    necessita.codice_art = articolo.codice;

end IF;
end
```

- **Before_Delete_Riceve**: Aggiorna la quantità di articoli in un ordine.
  Se stato ordine è “non accettato” o “accettato” aggiorna la quantità di articoli in ordine.
  Se stato ordine è “in esecuzione” o “terminato” non è possibile eliminarlo.
  Questo comporta che tutti gli Ordini_C devono restare memorizzati sulla base di dati.

```
CREATE TRIGGER `before_delete_riceve` BEFORE
  DELETE
  ON `riceve` FOR EACH row begin IF EXISTS
         (
                SELECT *
                FROM   ordine_c
                WHERE  old.codice_ord_c = ordine_c.codice
                AND    ordine_c.stato = "accettato" ) THEN
  UPDATE ordine_c
  SET    ordine_c.qnt_art = ordine_c.qnt_art - old.qnt
  WHERE  old.codice_ord_c = ordine_c.codice;

ELSEIF
  EXISTS
  (
         SELECT *
         FROM   ordine_c
         WHERE  old.codice_ord_c = ordine_c.codice
         AND    ordine_c.stato="non accettato" ) THEN
```

```
  UPDATE ordine_c
  SET    ordine_c.qnt_art = ordine_c.qnt_art - old.qnt
  WHERE  old.codice_ord_c = ordine_c.codice;

ELSEIF
  EXISTS
  (
         SELECT *
         FROM   ordine_c
         WHERE  old.codice_ord_c = ordine_c.codice
         AND    ordine_c.stato="in esecuzione" ) THEN
  signal SQLSTATE '45000' SET message_text = 'Ordine in esecuzione!';
ELSEIF
  EXISTS
  (
         SELECT *
         FROM   ordine_c
         WHERE  old.codice_ord_c = ordine_c.codice
         AND    ordine_c.stato="terminato" ) THEN
  signal SQLSTATE '45000' SET message_text = 'Ordine terminato!';
end IF;
end
```

- **Before_Insert_Ordine_C**: Imposta lo stato ordine ad “non accettato” e la quantità iniziale di articoli a 0.

```
CREATE TRIGGER `before_insert_ordine_c` BEFORE INSERT
ON `ordine_c`
FOR EACH row
begin
  SET new.stato = "non accettato";
  SET new.qnt_art = 0;
end
```

- **Before_Update_Ordine_C**: Aggiorna lo stato articolo quando cambia lo stato di Ordine_C.

```
CREATE TRIGGER `before_update_ordine_c` BEFORE UPDATE
ON `ordine_c`
FOR EACH row
begin
  IF ( old.stato != new.stato ) THEN
    IF ( new.stato = 'Accettato' )THEN
      UPDATE riceve
      SET    riceve.stato_art = 'in coda'
      WHERE  riceve.codice_ord_c = old.codice;
    ELSEIF ( new.stato = 'in esecuzione' )THEN
```

```
      UPDATE riceve
      SET    riceve.stato_art = 'in esecuzione'
      WHERE  riceve.codice_ord_c = old.codice;
    ELSEIF ( new.stato = 'terminato' )THEN
      UPDATE riceve
      SET    riceve.stato_art = 'finito'
      WHERE  riceve.codice_ord_c = old.codice;
    end IF;
  end IF;
end
```

- **Before_Delete_Ordine_C**: Segnala errore se lo stato dell'ordine è “non accettato” o “terminato”.

```
CREATE TRIGGER `before_delete_ordine_c` BEFORE
  DELETE
  ON `ordine_c` FOR EACH row begin IF (old.stato = 'in esecuzione')THEN signa
l SQLSTATE '45000' SET message_text = 'Ordine in esecuzione!';

ELSEIF
  (old.stato = 'terminato')THEN
  signal SQLSTATE '45000' SET message_text = 'Ordine terminato!';
end IF;
end
```

- **Before_Delete_Ordine_O:** Aggiorna le materie prime sottraendone la quantità di richiede.

```
CREATE TRIGGER `before_delete_ordine_o` BEFORE DELETE
ON `ordine_o`
FOR EACH row
begin
  UPDATE materie_prime,
         richiede,
         fornitore
  SET    materie_prime.qnt = materie_prime.qnt - richiede.qnt
  WHERE  richiede.codice_ord_o = old.codice
         AND richiede.codice_for = fornitore.codice_fiscale
         AND materie_prime.codice = fornitore.codice_mat;
end
```

- **After_Update_Ciclo**: Aggiorna il tempo di produzione dell'articolo nel caso cambi il tempo di pulizia o preparazione del ciclo.

```
CREATE TRIGGER `after_update_ciclo` after UPDATE
ON `ciclo`
FOR EACH row
begin
  IF ( new.tempo_prp != old.tempo_prp ) THEN
    UPDATE articolo
    SET    articolo.tempo_prd = Time (tempo_prd + ( new.tempo_prp -
                                                    old.tempo_prp ))
    WHERE  new.codice_art = articolo.codice;
  end IF;
  IF ( new.tempo_plz != old.tempo_plz ) THEN
    UPDATE articolo
    SET    articolo.tempo_prd = Time (tempo_prd + ( new.tempo_plz -
                                                    old.tempo_plz ))
    WHERE  new.codice_art = articolo.codice;
  end IF;
end
```

- **Before_Insert_Ciclo**: Aggiorna il tempo di produzione articolo e aggiorna il tempo di preparazione del ciclo se il ciclo è legato con qualche utensile altrimenti lo setta a 0.

```
CREATE TRIGGER `before_insert_ciclo` BEFORE INSERT
ON `ciclo`
FOR EACH row
begin
  SET new.tempo_prp = time('0:0:0');
  IF EXISTS (SELECT *
             FROM   articolo,
                    ciclo
             WHERE  articolo.codice = new.codice_art)THEN
    UPDATE articolo
    SET    articolo.tempo_prd = ( articolo.tempo_prd + Time (new.tempo_plz) )

    WHERE  articolo.codice = new.codice_art;
  ELSE
    UPDATE articolo
    SET    articolo.tempo_prd = Time (new.tempo_plz)
    WHERE  articolo.codice = new.codice_art;
  end IF;
end
```

- **Before_Delete_Ciclo**: Elimino anche l'articolo associato a quel ciclo.

```
CREATE TRIGGER `before_delete_ciclo` BEFORE DELETE
ON `ciclo`
FOR EACH row
begin
  DELETE FROM articolo
  WHERE  ( old.codice_art = articolo.codice );
end
```

- **After_Insert_Utensili**: Aggiunge il nuovo tempo di preparazione di un utensile al ciclo.

```
CREATE TRIGGER `after_insert_utensile` after INSERT
ON `utensile`
FOR EACH row
begin
  IF EXISTS(SELECT *
            FROM   utensile,
                   ciclo
            WHERE  ciclo.codice = new.codice_ccl)THEN
    UPDATE ciclo
    SET    ciclo.tempo_prp = Time(ciclo.tempo_prp + new.tempo_prp)
    WHERE  new.codice_ccl = ciclo.codice;
  end IF;
end
```

- **After_Update_Utensili**: Aggiorna il tempo di preparazione del ciclo.

```
CREATE TRIGGER `after_update_utensile` after UPDATE
ON `utensile`
FOR EACH row
begin
  IF ( old.codice_ccl <> new.codice_ccl ) THEN
    UPDATE ciclo
    SET    ciclo.tempo_prp = Time(ciclo.tempo_prp + old.tempo_prp)
    WHERE  new.codice_ccl = ciclo.codice;
  ELSEIF EXISTS( SELECT * FROM utensile, ciclo WHERE ciclo.codice =
  new.codice_ccl
  )THEN
    UPDATE ciclo
    SET    ciclo.tempo_prp = Time(ciclo.tempo_prp + (
                                  new.tempo_prp - old.tempo_prp ))
    WHERE  new.codice_ccl = ciclo.codice;
  end IF;
end
```

- **Before_Update_Esegue**: Aggiorna il tempo di produzione dell'articolo

```
CREATE TRIGGER `before_update_esegue` BEFORE
  UPDATE
  ON `esegue` FOR EACH row begin IF EXISTS
        (
               SELECT *
               FROM   articolo,
                      ciclo,
                      fase,
                      operazioni,
                      esegue
               WHERE  articolo.codice = ciclo.codice_art
               AND    ciclo.codice = fase.codice_ccl
               AND    fase.codice_operazioni = operazioni.codice
               AND    operazioni.codice = new.codice_operazioni ) THEN
  UPDATE articolo,
         ciclo,
```

```
        fase,
        operazioni,
        esegue
  SET   articolo.tempo_prd = Time (articolo.tempo_prd + (new.durata - old.durata ))
  WHERE articolo.codice = ciclo.codice_art
  AND   ciclo.codice = fase.codice_ccl
  AND   fase.codice_operazioni = operazioni.codice
  AND   operazioni.codice = new.codice_operazioni;

ELSE
  signal SQLSTATE '45000' SET message_text = 'Nessun articolo che esegue questa opera
zione';
end IF;
end
```

- **Before_Insert_Esegue**: Aggiunge il nuovo tempo di esecuzione dell'operazione al tempo di produzione dell'articolo.

```
CREATE TRIGGER `before_insert_esegue` BEFORE
  INSERT
  ON `esegue` FOR EACH row begin IF EXISTS
        (
              SELECT *
              FROM   articolo,
                     ciclo,
                     fase,
                     operazioni,
                     esegue
              WHERE  articolo.codice = ciclo.codice_art
              AND    ciclo.codice = fase.codice_ccl
              AND    fase.codice_operazioni = operazioni.codice
              AND    operazioni.codice = new.codice_operazioni
        )
        THEN
  UPDATE articolo
  SET    articolo.tempo_prd = Time (articolo.tempo_prd + new.durata);

ELSE
  signal SQLSTATE '45000' SET message_text = 'Nessun articolo che esegue ques
ta operazione';
end IF;
end
```

- **Before_Insert_Articolo**: imposta a 0 il tempo di produzione in quanto è un dato derivato.

```
CREATE TRIGGER `before_insert_articolo` BEFORE INSERT
ON `articolo`
FOR EACH row
begin
  SET new.tempo_prd = time(0);
end
```

- **Before_Insert_Fase**: Gestisce la progressione del numero della fase, ogni volta che viene inserita una fase ad un ciclo conta quanti cicli sono già associati e inserisce il valore contato più uno al valore del numero. Inoltre aggiorna il tempo di produzione dell'articolo aggiungendo il valore della durata dell'operazione contenuta nella tabella di esegue.

```
CREATE TRIGGER `before_insert_fase` BEFORE INSERT
ON `fase`
FOR EACH row
begin
  DECLARE n VARCHAR(10);
  SET n = (SELECT count(*) FROM fase WHERE fase.codice_ccl = new.codice_ccl)
+1;
  SET new.num = n;
  UPDATE articolo,
         ciclo,
         operazioni,
         esegue,
         fase
  SET    articolo.tempo_prd = Addtime(articolo.tempo_prd, esegue.durata)
  WHERE  articolo.codice = ciclo.codice_art
         AND ciclo.codice = new.codice_ccl
         AND operazioni.codice = new.codice_operazioni
         AND esegue.codice_operazioni = operazioni.codice;
end
```

- **Before_Update_Materie_Prime**: Quando la quantità arriva al di sotto della soglia viene inserito automaticamente un ordine creato da un operatore di default con codice “XXXXXXXXXXXXXXXX” alla data corrente e tra i fornitori sceglie quello che vende le materie prime con il prezzo più basso e ordina una quantità di materie prime pari alla soglia*10.

```
CREATE TRIGGER `before_update_materie_prime` BEFORE UPDATE
ON `materie_prime`
FOR EACH row
begin
  DECLARE cod CHAR(4);
  DECLARE cod_forn CHAR(16);
  SET cod = (SELECT max(ordine_o.codice) FROM ordine_o)+1;
  SET cod_forn = (SELECT fornitore.codice_fiscale FROM fornitore WHERE
  fornitore.codice_mat = old.codice AND fornitore.prezzo = ( SELECT
  min(f.prezzo) FROM
  fornitore f WHERE f.codice_mat = old.codice));
  IF( new.qnt < old.soglia )THEN
    INSERT INTO ordine_o
    VALUES      (cod,
                 'XXXXXXXXXXXXXXXX',
                 CURRENT_DATE);
    INSERT INTO richiede
    VALUES      (cod_forn,
                 cod,
                 old.soglia * 10);
    SET new.qnt = new.qnt + (old.soglia*10);
  end IF;
end
```

Op b) Dati Derivati

Possibili dati derivati:

- **qnt_articoli** in **Ordice_C** è la somma degli articoli in un ordine è conveniente?
- **tempo_prp** in **Ciclo** è la somma dei tempi di preparazione in Utensili conviene?
- **tempo_prd** in **Articolo** è la somma dei tempi di preparazione utensili, del tempo di pulizia del ciclo e della durata di ogni operazione nel ciclo.

Supposizioni generali:
Dal testo non si deduce una frequenza dei dati ma è descritta una quantità di tuple già presenti sul database. Per la discussione degli attributi derivati si è scelto e stimato che i dati forniti dal testo sono su base annua, di conseguenza avremo 16.000 articoli all'anno, 400 clienti all'anno, 8.000 ordini all'anno ecc... . Questa frequenza è stata calcolata immaginando una piccola ditta di provincia che opera su scala comunale.

Discussione qnt_articoli:

Supposizioni:

Ogni ordine ha in media due articoli.

Ogni volta che si crea un ordine si richiede la di visualizzare la quantità totale di articoli ordinati.

Un cliente si sbaglia in media 1 volta su 10 su un ordine effettuato, quindi provoca una modifica.

Tabella volumi

Concetto	Tipo	Volume
Articolo	E	16.000
Ordine	E	8.000
Riceve	A	16.000

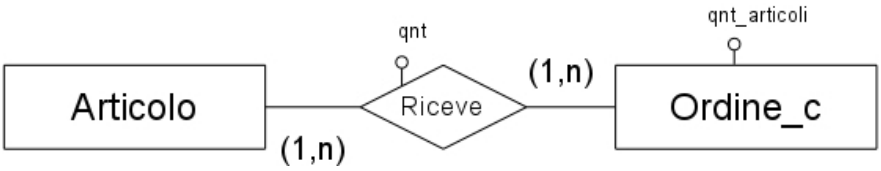


Tabella Operazioni

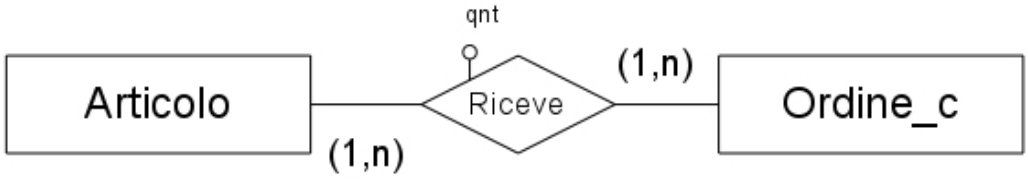
Operazione	Tipo	Frequenza	Descrizione
Op. 1	I	8.000/Anno	Lettura qnt_articoli
Op. 2	I	16.000/A	Aggiunta Ordine con articolo già presente
Op. 3	I	1.600/A	Modifica ordine

Tabella Calcoli con dato derivato

Op	Concetto	Accesso	Tipo	Totale
1	Ordine	1	Lettura	1 x 8.000 = 8.000 accessi/anno
2	Ordine Riceve Articolo	1 2 2	Scrittura Scrittura Lettura	8 x 16.000 = 128.000 accessi/anno
3	Ordine Ordine Riceve Riceve	1 1 2 2	Lettura Scrittura Lettura Scrittura	9 x 1.600 = 14.400 accessi/anno

Tot = 150.400 accessi/anno ovvero 0,00477 accessi/secondo.

Tabella Calcoli senza dato derivato



Op	Concetto	Accesso	Tipo	Totale
1	Ordine Riceve	1 2	Lettura Lettura	3 x 8.000 = 24.000 accessi/anno
2	Ordine	1	Scrittura	8 x 16.000 = 128.000

	Riceve Articolo	2 2	Scrittura Lettura	accessi/anno
3	Ordine Riceve Riceve	1 2 2	Lettura Lettura Scrittura	7 x 1.600 = 11.200 accessi/anno

Tot = 163.200 accessi/anno ovvero 0,00516 accessi/secondo.

Discussione tempo_prp in Ciclo:
Supposizioni:

Ad ogni creazione di un articolo si legge il tempo necessario alla preparazione dell'utensile. Ogni utensile viene riutilizzato almeno in 10 cicli di lavoro prima che si distrugga, pertanto la frequenza
di aggiunta di un nuovo utensile è di circa 1.600/anno.

Tabella volumi

Concetto	Tipo	Volume
Ciclo	E	16.000
Utensile	E	48.000
Possiede	A	48.000

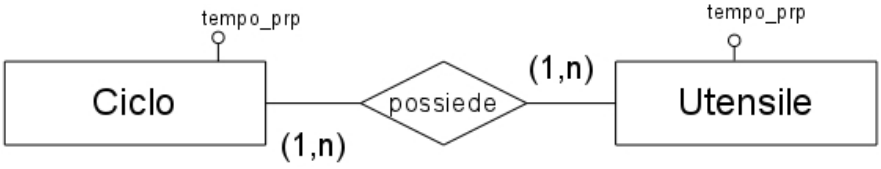


Tabella Operazioni

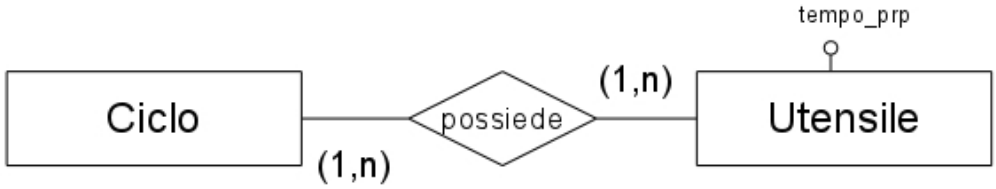
Operazione	Tipo	Frequenza	Descrizione
Op. 1	I	16.000/A	Lettura tempo di preparazione di tutti gli utensili
Op. 2	I	16.000/A	Aggiunta Utensile

Tabella Calcoli con dato derivato

Op	Concetto	Accesso	Tipo	Totale
1	Ciclo	1	Lettura	1 x 16.000 = 16.000 accessi/anno
2	Utensili Possiede Ciclo Ciclo	1 1 1 1	Scrittura Scrittura Lettura Scrittura	7 x 1.600 = 11.200 accessi/anno

Tot = 28.200 accessi/anno ovvero 0,00089 accessi/secondo.

Tabella Calcoli senza dato derivato



Op	Concetto	Accesso	Tipo	Totale
1	Ciclo Possiede Utensili	1 3 3	Lettura Lettura Lettura	7 x 16.000 = 112.000 accessi/anno

Tot = solo con l'operazione uno il costo senza dato derivato è maggiore perciò non continuiamo i calcoli.

Discussione tempo_prd in Articolo:

Supposizioni:
Ogni ciclo è composto in media da 3 fasi.
Ogni operazione ha associato in media 2 fasi.

Ogni macchina esegue in media 3 operazioni.
Si cambiano in media 1 su 20 macchine all'anno.

Tabella volumi

Concetto	Tipo	Volume
Articolo	E	16.000
Composto 1	A	16.000
Ciclo	E	16.000
Composto 2	A	48.000
Fase	E	48.000
E' in	A	48.000
Operazioni	E	24.000
Esegue	A	24.000
Macchine	E	8.000

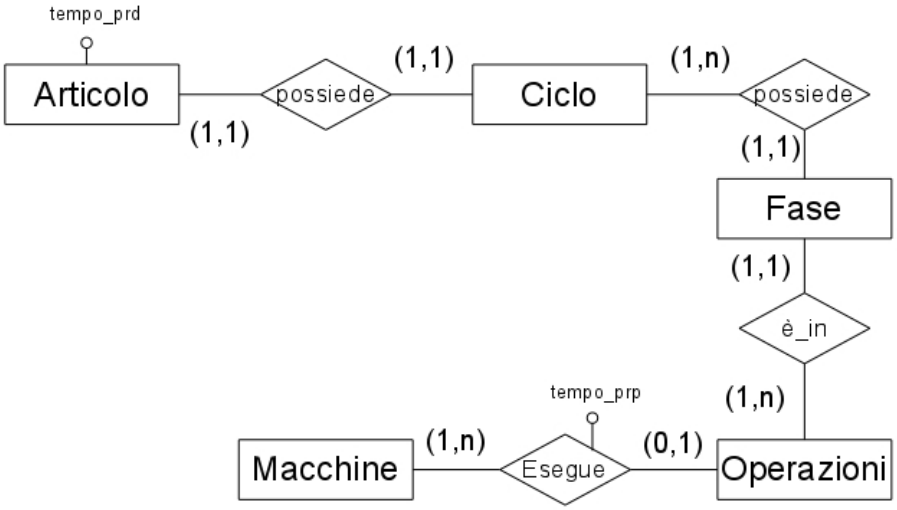


Tabella Operazioni

Operazione	Tipo	Frequenza	Descrizione
Op. 1	I	16.000/A	Lettura tempo produzione di un articolo
Op. 2	I	400/A	Aggiunta Macchina

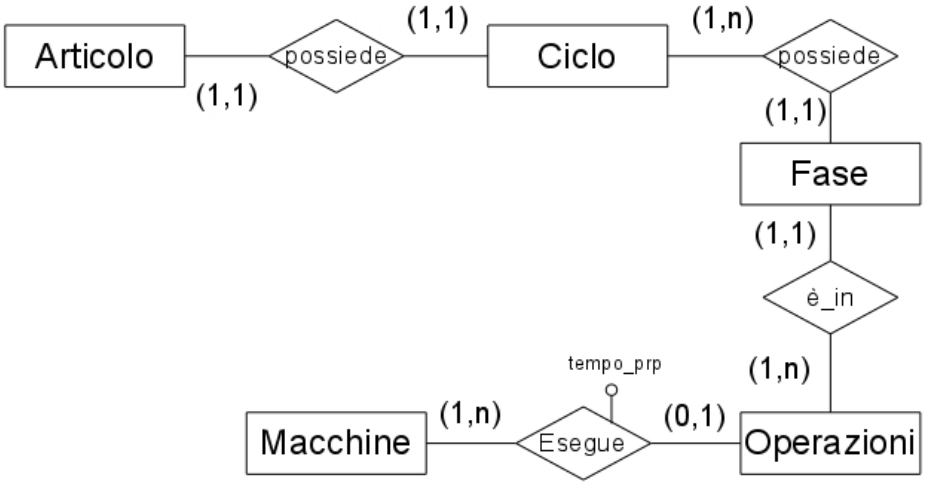
Tabella Calcoli con dato derivato

Op	Concetto	Accesso	Tipo	Totale
1	Articolo	1	Lettura	1 x 16.000 = 16.000 accessi/anno
2	Macchina	1	Scrittura	39 x 400 = 15.600 accessi/anno
	Esegue	3	Scrittura	
	Operazione	3	Lettura	
	E' in	6	Lettura	
	Fase	6	Lettura	
	Composto2	6	Lettura	

	Ciclo	6/3	Lettura	
	Composto1	6/3	Lettura	
	Articolo	6/3	Lettura	
	Articolo	6/3	Scrittura	

Tot = 31.600 accessi/anno ovvero 0,00100 accessi/secondo.

Tabella Calcoli senza dato derivato



Op	Concetto	Accesso	Tipo	Totale
1	Articolo	1	Lettura	14 x 16.000 = 224.000 accessi/anno
	Composto1	1	Lettura	
	Ciclo	1	Lettura	
	Composto2	3	Lettura	
	Fase	3	Lettura	
	E' in	3	Lettura	
	Operazioni	3/2	Lettura	
	Esegue	3/2	Lettura	

Tot = solo con l'operazione uno il costo senza dato derivato è maggiore perciò non continuiamo i calcoli.

Discussione qnt_occ:

Supposizioni:

Un cliente si sbaglia in media 1 volta su 10 su un ordine effettuato, quindi provoca una modifica.

Tabella volumi

Concetto	Tipo	Volume
Articolo	E	16.000
Materie_Prime	E	400
Necessita	A	6.400.000
Riceve	A	128.000.000

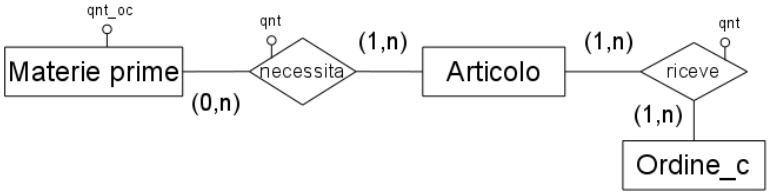


Tabella Operazioni

Operazione	Tipo	Frequenza	Descrizione
Op. 1	I	16.000/A	Lettura della quantità di materie occupata da un articolo
Op. 2	I	8.000/A	Modifica dello stato di un ordine

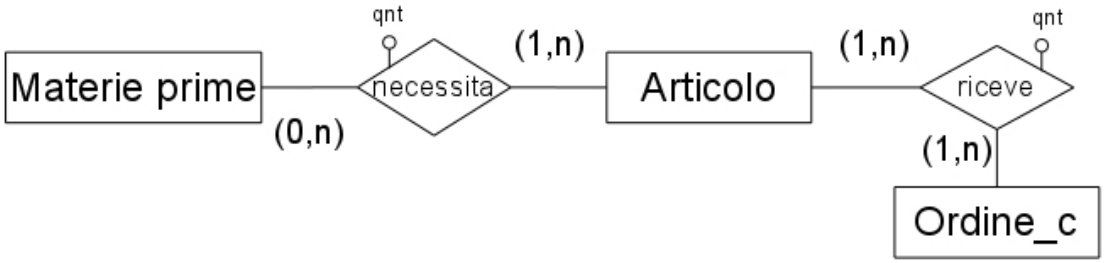
Tabella Calcoli con dato derivato

Op	Concetto	Accesso	Tipo	Totale
1	Materie_prime	1	Lettura	1 x 16.000 = 16.000 accessi/anno
2	Ordine	1	Lettura	325 x 8000 = 2.600.000 accessi/anno
	Riceve	2	Lettura	
	Articolo	2	Lettura	
	Necessita	80	Lettura	
	Materie_prime	80	Lettura	

	Materie_prime	80	Scrittura	

Tot= 2.616.000 ovvero 0,085 accessi/secondo

Tabella Calcoli senza dato derivato



Op	Concetto	Accesso	Tipo	Totale
1	Materie_prime	1	Lettura	
	Necessita	40	Lettura	
	Articolo	40	Lettua	141 x 16.000 = 3.856.000
	Riceve	80	Lettura	
	Ordine	80	Lettura	

Tot = solo con l'operazione uno il costo senza dato derivato è maggiore perciò non continuiamo i calcoli.

Conclusione
Come si può dedurre dai calcoli tutti gli dati derivati sono stati mantenuti in quanto contribuiscono a velocizzare le operazioni.