UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI SCIENZE MM. FF. NN.

INFORMATICA

GRM - GESTORE RICHIESTE DI MANUTENZIONE

BASI DI DATI

Autori:

Cesarato Fabio

1019749

Sturaro Agostino

612088

Anno Accademico:

2010-2011

Indice

Sommario

GESTIONE RICHIESTE DI MANUTENZIONE

ABSTRACT

GRM (Gestore Richieste di Manutenzione) è un trouble ticketing system, un sistema che automatizza l'immissione, lo smistamento e la risoluzione di richieste, rivolto alla piccola industria metalmeccanica.
Come principali finalità, GRM si propone di facilitare e velocizzare la gestione delle manutenzioni, permettendone una storicizzazione.
Il bacino di utenza si ripartisce in due grandi categorie: gli operatori, della linea di produzione, ed i manutentori, solitamente in Officina.
Un operatore può utilizzare GRM per segnalare la presenza di un problema relativo ad un macchinario. I manutentori possono visualizzare tali segnalazioni e valutare se intervenire, nel caso la situazione lo richieda, o scartare la segnalazione nel caso il problema non sussista.

ANALISI DEI REQUISITI

AMBITO DI UTILIZZO

GRM è una interfaccia web il cui scopo è permettere una semplice ed intuitiva gestione delle richieste di manutenzione. È rivolto a piccole/medie aziende metalmeccaniche, in cui vi sono degli operatori impiegati in una generica linea di produzione e dei manutentori pronti a risolvere possibili guasti.
L'operatore potrà segnalare il malfunzionamento di una macchina direttamente dal postazione di lavoro, mediante un apposito terminale connesso alla rete. Tale segnalazione genererà una richiesta di manutenzione che verrà immediatamente resa visibile ai manutentori. Sarà quindi uno di loro a valutare la richiesta e decidere di aprire una manutenzione o di annullare la richiesta in quanto errata.
Le richieste in attesa o non totalmente evase verranno costantemente presentate ai manutentori. Ogni richiesta ed ogni manutenzione verrà registrata e mantenuta in modo permanente. Il sistema permette quindi di mantenere uno storico di tutte le richieste e delle manutenzioni effettuate, permettendo inoltre di ottenete informazioni di dettaglio.

INDICE DELLE CLASSI

- Utenti
- Dipendenti
- Operatori
- Manutentori
- Segretari
- Reparti
- Macchine

- Richieste di Manutenzione
- Stati Richieste
- Interventi di manutenzione
- Ricambi

Utenti

Sono gli account dei *dipendenti*.
Di un utente interessano:

- Codice univoco di identificazione
- Password
- Se e' stato attivato
- Massimo numero di risultati visualizzabili in una pagina

Un account identifica uno specifico *dipendente*.
Un utente viene creato dal dipendente quando si registra, e viene attivato da un *segretario*.

Dipendenti

Sono le persone che lavorano nell'azienda.
Di un dipendente interessano:

- Codice univoco di identificazione
- Nome
- Cognome
- Reparto in cui lavora
- Data di assunzione
- Data di fine rapporto
- Codice Fiscale
- Retribuzione Oraria
- Sesso

Ogni dipendente è associato ad uno specifico *reparto*, e può essere assunto con la mansione di *manutentore, operatore* o *segretario*.
La retribuzione oraria può essere maggiore o uguale a zero, ma deve essere sempre definita.
Ogni dipendente ha un proprio account *utente*, di cui può cambiare la password ed il massimo numero di risultati visualizzabili in una pagina.

Operatori

Sono un tipo di *dipendenti*.
Gli operatori sono abilitati a generare *richieste di manutenzione* nel caso di eventuali malfunzionamenti delle *macchine* del loro *reparto.*

Manutentori

Sono un tipo di *dipendenti*.
I manutentori sono abilitati ad accettare *richieste di manutenzione*, ad effettuare *interventi di manutenzione*, nei quali possono utilizzare una numero indefinito di *ricambi*, e, alla fine di un intervento, chiudere la richiesta quando la macchina ritorna ad essere pienamente operativa.

Segretari

Sono un tipo di *dipendenti*.
I segretari possono:

- accettare nuovi utenti che richiedono di accedere al sistema
- effettuare operazioni di anagrafica, in particolare inserimento o eliminazione di: Reparti, Macchine, Ricambi e Utenti
- aggiornare la retribuzione di un qualsiasi Dipendente (ad eccezione di se stessi)
- cambiare la password di un qualsiasi Dipendente (ad eccezione di se stessi)
- cercare una richiesta con parametri a loro discrezione, avendo a disposizione una dettagliata descrizione
- accedere ad informazioni di analisi sui dati, come: costi e tempi delle varie richieste

Reparti

Sono le aree di lavoro che si vogliono identificare all'interno dell'azienda.
Di un reparto interessano:

- Codice univoco di identificazione
- Breve descrizione

In un reparto possono trovarsi un numero variabile di *dipendenti* e di *macchine*, talvolta anche nessuno.

Macchine

Sono i macchinari e gli utensili utilizzati nell'azienda.
Di una macchina interessano:

- Codice univoco di identificazione
- Breve descrizione
- Reparto in cui è posizionata

Le macchine sono il soggetto delle *richieste di manutenzione*.

Richieste di manutenzione

Sono le segnalazioni digitali il cui scopo è presentare ai *manutentori* il presunto malfunzionamento di una *macchina*.
Di una richiesta di manutenzione interessano:

- Codice univoco di identificazione
- Descrizione del problema riscontrato sulla macchina
- Operatore che l'ha generata
- Codice dello Stato (informazione sullo stadio di lavorazione)
- Codice della macchina cui si riferisce
- Data ed ora in cui è stata generata
- Data ed ora in cui è stata accettata
- Data ed ora in cui è stata evasa

Possono essere generate soltanto dagli *operatori*, e processate soltanto dai *manutentori*.
Vi può essere solo una richiesta aperta per macchina. Per aprirne un'altra quella aperta deve essere chiusa.

Ogni *richiesta di manutenzione* si trova sempre in uno *stato* definito.
Un richiesta può essere annullata solo dall'operatore che l'ha generata, ma può essere rifiutata da un qualsiasi manutentore di cui non interessa conoscere l'identità.
Se la richiesta non viene rifiutata o annullata, sarà soggetta ad almeno un intervento di manutenzione prima di essere evasa.

Stati richieste

Sono le possibili fasi in cui una *richiesta di manutenzione* può trovarsi.
Dello stato di una richiesta interessano:

- Codice univoco di identificazione
- Breve descrizione

I possibili stati associati ad una richiesta sono:

- Attesa, richiesta appena generata e ancora da processare
- Annullata, richiesta annullata dall'operatore che l'ha generata, presumibilmente per errore, prima di essere processata
- Rifiutata, richiesta considerata non valida da un manutentore
- In Esecuzione, richiesta per cui e' stato aperto almeno un intervento
- Eseguita, richiesta per cui l'ultimo intervento ha risolto completamente il problema

Le transizioni possibili tra gli stati di una *richiesta* sono:

- da Attesa ad Annullato, Rifiutato o In Esecuzione
- da In Esecuzione ad Eseguita

Ricambi

Sono i materiali utilizzati durante un *intervento di manutenzione*.
Di un ricambio interessano:

- Codice univoco di identificazione
- Breve descrizione
- Costo

Un ricambio viene inteso come una categoria, e non come lo specifico oggetto impiegato.

Interventi di manutenzione

Sono le operazioni eseguite dai *manutentori* per riportare una *macchina* ad essere operativa.
Un solo intervento non necessariamente riporta la macchina all'operatività.
Un manutentore può generare più interventi di manutenzione, non contemporanei, riferiti ad una richiesta.
Di un intervento di manutenzione interessano:

- Codice univoco di identificazione
- Codice della richiesta a cui è associato
- Manutentore che la effettua
- Ricambi utilizzati
- Data ed ora in cui inizia
- Data ed ora in cui finisce

In un *intervento di manutenzione* vengono utilizzati un certo numero di *ricambi*, anche nessuno. Di ogni ricambio può essere impiegato un numero variabile di pezzi.

PROGETTAZIONE CONCETTUALE

CLASSI

- Utenti:
  - Username: string <<PK>>
  - PasswordHash: string <<not null>>
  - Attivo: bool <<not null>>
  - RisultatiPerPagina: int <<not null>>

- Dipendenti:
  - IdDipendente: int <<PK>>
  - Nome: string <<not null>>
  - Cognome: string <<not null>>
  - DataAssunzione: date <<not null>>
  - DataFineRapporto: date
  - CF: string
  - Sesso: enum(M, F) <<not null>>
  - RetribuzioneOraria: float <<not null>>

    Vincoli:
    - RetribuzioneOraria deve essere maggiore di 0

- Operatori:
  - nessun attributo proprio

- Manutentori:
  - nessun attributo proprio

- Segretari:
  - nessun attributo proprio

- Reparti:
  - CodReparto: int <<PK>>
  - DescrReparto: string

- Macchine:

- CodMacchina: int <<PK>>
- DescrMacchina: string

- Richieste di Manutenzione
  - IdRichiesta: int <<PK>>
  - DescrRichiesta: string
  - DataGenerazione: datetime <<not null>>
  - DataEvasione: datetime

    Vincoli:
    - Vi può essere una sola richiesta aperta per macchina
    - Le transizioni di Stato permesse sono: da In Attesa a Annullata, Rifiutata o In Esecuzione; da In Esecuzione ad Eseguita
    - La Data di Evasione deve essere maggiore di quella di Generazione

- Stati Richieste
  - CodStato: string <<PK>>
  - Descrizione: string <<not null>>

- Interventi di Manutenzione
  - CodInterManut: int <<PK>>
  - DataInizio: datetime <<not null>>
  - DataFine: datetime

- Ricambi
  - CodRicambio: int <<PK>>
  - DescrRicambio: string
  - CostoPezzo: float

ASSOCIAZIONI

Utenti-Dipendenti: IdentificaUn

- Un utente identifica un singolo dipendente
  Un dipendente è identificato da un singolo utente
- Molteplicità: 1:1
- Totalità:
  - Totale da Utenti a Dipendenti
  - Parziale da Dipendenti a Utenti

Macchine-Reparti: PosizionataIn

- Una macchina è posizionata in un singolo reparto
  In un reparto sono posizionate una, nessuna o più macchine
- Molteplicità: 1:N
- Totalità:
  - Totale da Macchine a Reparti
  - Parziale da Reparti a Macchine

Dipendenti-Reparti: LavoraIn

- Un dipendente lavora in un singolo reparto
  In un reparto lavorano uno, nessuno o più dipendenti
- Molteplicità: 1:N
- Totalità:
  - Parziale da Reparti a Operatori
  - Totale da Dipendenti a Reparti

Operatori-Richieste di Manutenzione: Genera

- Un operatore generare una, nessuna o più richieste di manutenzione
  Una richiesta di manutenzione è generata da un singolo operatore
- Molteplicità: N:1
- Totalità:
  - Parziale da Operatori a Richieste di Manutenzione
  - Totale da Richieste di Manutenzione a Operatori

Richieste di Manutenzione-Macchine: EffettuataSu

- Una richiesta di manutenzione è effettuata su una singola macchina
  Su una macchina vengono effettuate una, nessuna o più richieste di manutenzione
- Molteplicità: 1:N
- Totalità:
  - Totale da Richieste di Manutenzione a Macchine
  - Parziale da Macchine a Richieste di Manutenzione

Richieste di Manutenzione-Stati Richieste: SiTrovaIn

- Una richiesta di manutenzione si trova in un singolo stato
  In uno stato si trovano una, nessuna o più richieste di manutenzione
- Molteplicità: 1:N
- Totalità:
  - Totale da Richieste di Manutenzione a Stati Richieste
  - Parziale da Stati Richieste a Richieste di Manutenzione

Interventi di manutenzione-Richieste di Manutenzione: AssociatoA

- Un intervento di manutenzione è associato a una singola richiesta di manutenzione

Ad una richiesta di manutenzione sono associati uno, nessuno o più interventi di manutenzione

- Molteplicità: 1:N
- Totalità:
    - Totale da Interventi di Manutenzione a Richieste di Manutenzione
    - Parziale da Richieste di Manutenzione a Interventi di manutenzione

Manutentori-Interventi di manutenzione: LavoraSu

- Un manutentore lavora su uno, nessuno o più interventi di manutenzione
  Su un intervento di manutenzione lavora un singolo manutentore
- Molteplicità: N:1
- Totalità:
    - Parziale da Manutentori a Interventi di manutenzione
    - Totale da Interventi di manutenzione a Manutentori

Interventi di manutenzione-Ricambi: RicambiUtilizzati

- Un intervento di manutenzione richiede uno, nessuno o più ricambi
  Un ricambio può essere usato in uno, nessuno o più interventi di manutenzione
- Per ogni tipo di ricambio utilizzato in un intervento interessa sapere il numero di pezzi
- Molteplicità: N:M
- Totalità:
    - Parziale da Interventi di manutenzione a Ricambi
    - Totale da Ricambi a Interventi di manutenzione

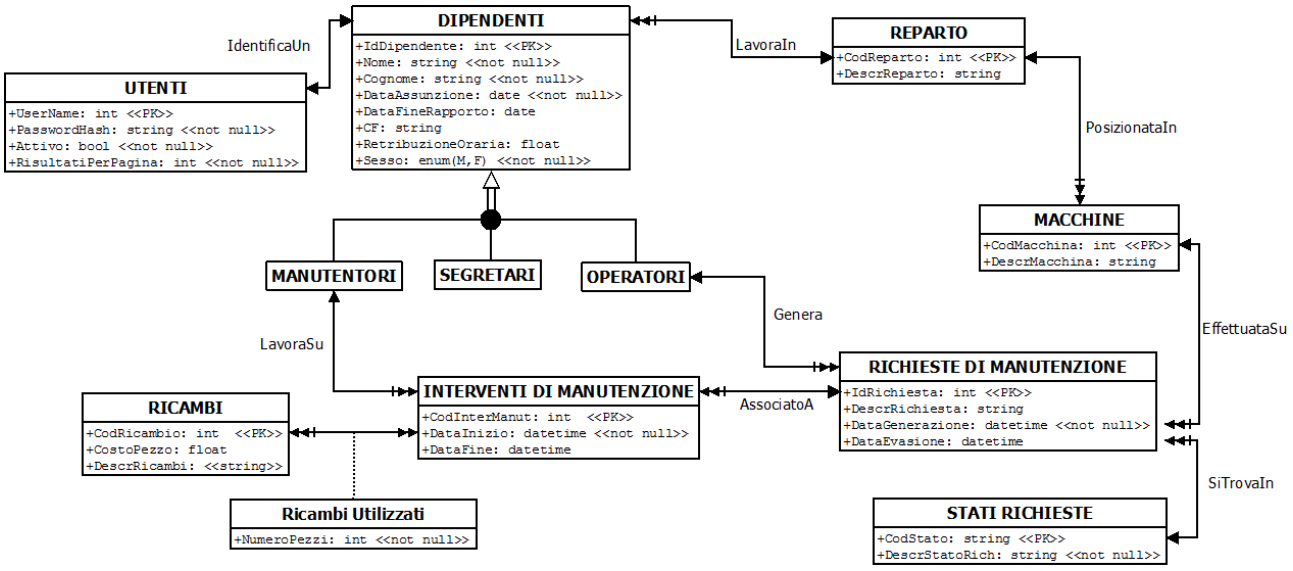
NOTE DI MODELLAZIONE

- Data ed ora di accettazione di una Richiesta vengono dedotte dalla data e ora di inizio del primo intervento associato
- Un nuovo utente di default viene creato come Non Attivato, verrà attivato successivamente da un segretario.
- Una nuova richiesta viene generata in Stato di Attesa.
- Un manutentore può avere un solo intervento aperto per volta
- I controlli sulle transizioni di stato vengono fatti mediante procedure SQL richiamate tramite PHP
- Per motivi di sicurezza invece della password viene salvato un suo hash
- I controlli su chi effettua le modifiche allo stato delle richieste vengono eseguiti a livello di interfaccia grafica

GERARCHIA DELLE CLASSI

Nello schema è presente la superclasse Dipendenti, dalla quale si originano tre sottoclassi: gli Operatori, i Manutentori ed i Segretari.

SCHEMA CONCETTUALE



PROGETTAZIONE LOGICA- RELAZIONALE

RAPPRESENTAZIONE ASSOCIAZIONI

Utenti-Dipendenti: IdentificaUn

Relazione 1:1, totale da Utenti a Dipendenti.

Sulla tabella Utenti viene aggiunta la chiave esterna:
IdDipendente <<unique>> <<not null>> <<FK(Dipendenti)>>

L'univocità dell'inversa di viene rappresentata con vincolo unique sulla chiave esterna.
Inoltre, dato che viene scelta la direzione della relazione in modo che essa sia totale, viene posto il vincolo not null sulla chiave esterna.

Macchine-Reparti: PosizionataIn

Relazione 1:N, totale da Macchine a Reparti.

Sulla tabella Macchine viene aggiunta la chiave esterna:
CodReparto <<not null>> <<FK(Reparti)>>

La totalità della relazione da Macchine a Reparti viene rappresentata mediante il vincolo not null sulla chiave esterna.

Dipendenti-Reparti: LavoraIn

Relazione 1:N, totale da Dipendenti a Reparti.

Sulla tabella Dipendenti viene aggiunta la chiave esterna:
CodReparto <<not null>> <<FK(Reparti)>>

La totalità della relazione da Dipendenti a Reparti viene rappresentata mediante il vincolo not null sulla chiave esterna.

Operatori-RichiesteManutenzione: Genera

Relazione 1:N, totale da RichiesteManutenzione a Operatori.

Sulla tabella RichiesteManutenzione viene aggiunta la chiave esterna:
IdOperatore <<not null>> <<FK(Operatori)>>

La totalità della relazione da RichiesteManutenzione a Operatori viene rappresentata mediante il vincolo not null sulla chiave esterna.

RichiesteManutenzione-Macchine: EffettuataSu

Relazione 1:N, totale da RichiesteManutenzione a Macchine.

Sulla tabella RichiesteManutenzione viene aggiunta la chiave esterna:
CodMacchina: int <<not null>> <<FK(Macchine)>>

La totalità della relazione da RichiesteManutenzione a Macchine viene rappresentata mediante il vincolo not null sulla chiave esterna.

RichiesteManutenzione-StatiRichieste: SiTrovaIn

Relazione 1:N, totale da RichiesteManutenzione a StatiRichieste.

Sulla tabella RichiesteManutenzione viene aggiunta la chiave esterna:
CodStato <<not null>> <<FK(StatiRichieste)>>

La totalità della relazione da RichiesteManutenzione a StatiRichieste viene rappresentata mediante il vincolo not null sulla chiave esterna.

InterventiManutenzione-RichiesteManutenzione: AssociatoA

Relazione 1:N, totale da Interventi di Manutenzione a Richieste di Manutenzione.

Sulla tabella InterventiManutenzione viene aggiunta la chiave esterna:
IdRichiesta <<not null>> <<FK(RichiesteManutenzione)>>

La totalità della relazione da InterventiManutenzione a RichiesteManutenzione viene rappresentata mediante il vincolo not null sulla chiave esterna.

Manutentori-InterventiManutenzione: LavoraSu

Relazione 1:N, totale da Interventi di manutenzione a Manutentori.

Sulla tabella InterventiManutenzione viene aggiunta la chiave esterna:
IdManutentore <<not null>> <<FK(Manutentori)>>

La totalità della relazione da InterventiManutenzione a Manutentori viene rappresentata mediante il vincolo not null sulla chiave esterna.

InterventiManutenzione-Ricambi: RicambiUtilizzati

Relazione N:M, totale da Ricambi a Interventi di manutenzione.

Per esplicitare tale relazione è necessario creare una tabella a cui viene assegnato il nome della relazione. La nuova tabella contiene come FK i puntatori alle tabelle che si vogliono mettere in relazione, tali puntatori divengono le chiavi PK della nuova tabella.
Gli attributi della relazione diventano attributi della nuova tabella creata.

Nuova Tabella:
- RicambiUtilizzati

Puntatori e Chiavi
- CodInterManut <<PK>> <<FK(InterventiManutenzione)>>
- CodRicambio <<PK>> <<FK(Ricambi)>>

Attributi:
- NumeroRicambi

GERARCHIE

Nello schema concettuale è presente la superclasse Dipendenti, dalla quale si originano tre sottoclassi: gli Operatori, i Manutentori ed i Segretari.
Operatori e Manutentori hanno delle associazioni proprie, per questo è sconsigliato rappresentarle mediante una tabella unica. Inoltre, anche la superclasse è parte un'associazione, e ciò rende sconsigliabile l'utilizzo del partizionamento orizzontale.
Si è quindi scelto di utilizzare il *partizionamento verticale* per rappresentare la gerarchia.
Utilizzando tale partizionamento viene perso il vincolo di disgiunzione tra le classi.

PROGETTAZIONE RELAZIONALE

- Utenti:
  - Username: string <<PK>>
  - PasswordHash: string <<not null>>
  - Attivo: bool <<not null>>
  - RisultatiPerPagina: int <<not null>>
  - IdDipendente: int <<not null>> <<unique>> <<FK(Dipendenti)>>

- Dipendenti:
  - IdDipendente: int <<PK>>
  - Nome: string <<not null>>
  - Cognome: string <<not null>>
  - CodReparto: int <<not null>> <<FK(Reparti)>>
  - DataAssunzione: date <<not null>>
  - DataFineRapporto: date
  - CF: string
  - RetribuzioneOraria: float con 2 numeri dopo la virgola
  - Sesso: enum(M, F) <<not null>>

- Manutentori:
  - IdManutentore: int <<PK>> <<FK(Dipendenti)>>

- Operatori:
  - IdOperatore: int <<PK>> <<FK(Dipendenti)>>

- Reparti:
  - CodReparto: int <<PK>>
  - DescrReparto: string

- Macchine:
  - CodMacchina: int <<PK>>
  - CodReparto: int <<not null>> <<FK(Reparti)>>
  - DescrMacchina: string

- RichiesteManutenzione
  - IdRichiesta: int <<PK>>
  - DescrRichiesta: string
  - IdOperatore: int <<not null>> <<FK(Operatori)>>
  - CodStato: string <<not null>> <<FK(StatiRichieste)>>
  - CodMacchina: int <<not null>> <<FK(Macchine)>>
  - DataGenerazione: datetime <<not null>>
  - DataEvasione: datetime

- StatiRichieste
  - CodStato: string <<PK>>
  - DescrStatoRich: string <<not null>>

- InterventiManutenzione
  - CodInterManut: int <<PK>>

- IdManutentore: int <<not null>> <<FK(Manutentori)>>
- IdRichiesta: int <<not null>> <<FK(RichiesteManutenzione)>>
- DataInizio: datetime <<not null>>
- DataFine: datetime

- Ricambi
  - CodRicambio: int <<PK>>
  - DescrRicambi
  - CostoPezzo: float con 2 numeri dopo la virgola

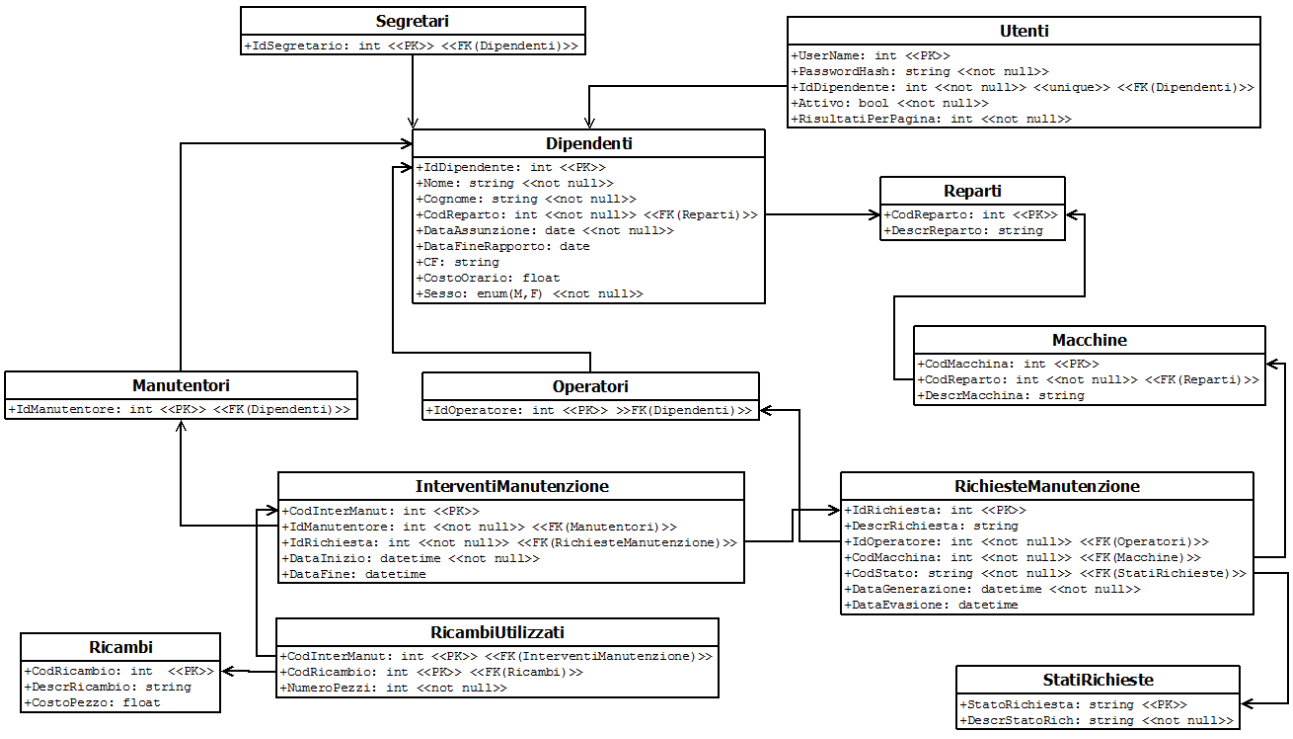
- RicambiUtilizzati
  - CodInterManut: int <<PK>> <<FK(InterventiManutenzione)>>
  - CodRicambio: int <<PK>> <<FK(Ricambi)>>
  - NumeroRicambi: int <<not null>>

  Vincoli:
    - NumeroRicambi dev'essere maggiore di zero

NOTE

- Il campo Sesso dei Dipendenti è un attributo di un tipo enumerativo. In linea di principio andrebbe tradotto in una nuova classe, non essendo offerto dal modello relazionale base. Però, dato che la maggior parte dei DBMS (MySQL incluso) lo implementa, è pratica comune mantenerlo anche nel relazionale, senza alcuna traduzione.

SCHEMA LOGICO



IMPLEMENTAZIONE SCHEMA LOGICO

Script SQL per la creazione delle tabelle SQL e l'inserimento di valori indispensabili.

```
/*SCRIPT CREAZIONE TABELLE*/

/*Disabilito le Foregin Key*/
SET FOREIGN_KEY_CHECKS = 0;

# -------------------- PULIZIA --------------------
/*Ripulisce, eliminando le tabelle qualora esistessero già*/
DROP TABLE IF EXISTS Dipendenti;
DROP TABLE IF EXISTS Utenti;
DROP TABLE IF EXISTS Manutentori;
DROP TABLE IF EXISTS Operatori;
DROP TABLE IF EXISTS Segretari;
DROP TABLE IF EXISTS Reparti;
DROP TABLE IF EXISTS Macchine;
DROP TABLE IF EXISTS StatiRichieste;
DROP TABLE IF EXISTS RichiesteManutenzione;
DROP TABLE IF EXISTS InterventiManutenzione;
DROP TABLE IF EXISTS Ricambi;
DROP TABLE IF EXISTS RicambiUtilizzati;

# -------------------- CREAZIONE --------------------

/*Creo la tabella Dipendenti*/
CREATE TABLE Dipendenti (
    IdDipendente INT AUTO_INCREMENT,
    Nome VARCHAR(30) NOT NULL,
    Cognome VARCHAR(30) NOT NULL,
    CodReparto INT NOT NULL,
    DataAssunzione DATE NOT NULL,
    DataFineRapporto DATE,
    CF CHAR(16),
    RetribuzioneOraria DECIMAL(10,2) NOT NULL DEFAULT 0,
    Sesso ENUM('M','F'),
    PRIMARY KEY pk_Dipendenti (IdDipendente),
    FOREIGN KEY fk_RepartoDipendente (CodReparto)
        REFERENCES Reparti (CodReparto)
        ON UPDATE CASCADE
    ) Engine = InnoDB;

/*Creo la tabella Utenti*/
CREATE TABLE Utenti (
    Username VARCHAR(15),
    PasswordHash CHAR(40) NOT NULL,
    IdDipendente INT NOT NULL UNIQUE,
    Attivo TINYINT(1) NOT NULL DEFAULT 0, /*su MySQL non esistono i booleani*/
    RisultatiPerPagina INT NOT NULL DEFAULT 15,
    PRIMARY KEY pk_Dipendenti (UserName),
    FOREIGN KEY fk_DipendentiUtenti (IdDipendente)
        REFERENCES Dipendenti (IdDipendente)
        ON UPDATE CASCADE
        ON DELETE CASCADE
    ) Engine = InnoDB;

/*Creo la tabella Manutentori*/
CREATE TABLE Manutentori (
    IdManutentore INT,
    PRIMARY KEY pk_Manutentori (IdManutentore),
    FOREIGN KEY fk_DipendenteManutentore (IdManutentore)
        REFERENCES Dipendenti (IdDipendente)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    ) Engine = InnoDB;

/*Creo la tabella Operatori*/
CREATE TABLE Operatori (
    IdOperatore INT,
    PRIMARY KEY pk_Operatori (IdOperatore),
    FOREIGN KEY fk_DipendenteOperarore (IdOperatore)
        REFERENCES Dipendenti (IdDipendente)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    ) Engine = InnoDB;
```

```
/*Creo la tabella Segretari*/
CREATE TABLE Segretari (
    IdSegretario INT,
    PRIMARY KEY pk_Segretari (IdSegretario),
    FOREIGN KEY fk_DipendenteSegretario (IdSegretario)
        REFERENCES Dipendenti (IdDipendente)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    ) Engine = InnoDB;

/*Creo la tabella Reparti*/
CREATE TABLE Reparti (
    CodReparto INT,
    DescrReparto VARCHAR (50),
    PRIMARY KEY pk_Reparti (CodReparto)
    ) Engine = InnoDB;

/*Creo la tabella Macchine*/
CREATE TABLE Macchine (
    CodMacchina INT,
    DescrMacchina VARCHAR (50),
    CodReparto INT NOT NULL,
    PRIMARY KEY pk_Macchine (CodMacchina),
    FOREIGN KEY fk_RepartoMacchina (CodReparto)
        REFERENCES Reparti (CodReparto)
        ON UPDATE CASCADE
    ) Engine = InnoDB;

/*Creo la tabella Stati Richieste*/
CREATE TABLE StatiRichieste(
    CodStato CHAR(3),
    DescrStatoRich VARCHAR(50) NOT NULL,
    PRIMARY KEY pk_StatiRichieste (CodStato)
    ) ENGINE = InnoDB;

/*Creo la tabella Richieste di Manutenzione*/
CREATE TABLE RichiesteManutenzione(
    IdRichiesta INT AUTO_INCREMENT,
    DescrRichiesta VARCHAR(500),
    IdOperatore INT NOT NULL,
    CodMacchina INT NOT NULL,
    CodStato CHAR(3) NOT NULL DEFAULT 'ATT', /*Di default la richiesta viene creata in Attesa*/
    DataGenerazione DATETIME NOT NULL,
    DataEvasione DATETIME,
    PRIMARY KEY pk_RichiesteManutenzione (IdRichiesta),
    FOREIGN KEY fk_OperatoreRichManut (IdOperatore)
        REFERENCES Operatori (IdOperatore)
        ON UPDATE CASCADE,
    FOREIGN KEY fk_MacchinaRichManut (CodMacchina)
        REFERENCES Macchine (CodMacchina)
        ON UPDATE CASCADE,
    FOREIGN KEY fk_StatoRichManut (CodStato)
        REFERENCES StatiRichieste (CodStato)
        ON UPDATE CASCADE
    ) ENGINE = InnoDB;

/*Creo la tabella Attività di Manutenzione*/
CREATE TABLE InterventiManutenzione(
    CodInterManut INT AUTO_INCREMENT,
    IdRichiesta INT NOT NULL,
    IdManutentore INT NOT NULL,
    DataInizio DATETIME NOT NULL,
    DataFine DATETIME,
    PRIMARY KEY pk_InterventiManutenzione (CodInterManut),
    FOREIGN KEY fk_RichManutInterventiManutenzione (IdRichiesta)
        REFERENCES RichiesteManutenzione (IdRichiesta)
        ON UPDATE CASCADE,
    FOREIGN KEY fk_ManutentoreInterventiManutenzione (IdManutentore)
        REFERENCES Manutentori (IdManutentore)
        ON UPDATE CASCADE,
    CHECK (DataFine>=DataInizio) /*Inutile su MySQL*/
    ) ENGINE = InnoDB;

/*Creo la tabella Ricambi*/
CREATE TABLE Ricambi(
    CodRicambio INT,
    DescrRicambio CHAR(50),
    CostoPezzo DECIMAL(10,2),
```

```
    PRIMARY KEY pk_Ricambi (CodRicambio)
    ) ENGINE = InnoDB;

/*Creo la tabella RicambiUtilizzati*/
CREATE TABLE RicambiUtilizzati(
    CodInterManut INT,
    CodRicambio INT,
    NumeroPezzi INT NOT NULL,
    PRIMARY KEY pk_RicambiUtilizzati (CodRicambio,CodInterManut),
    FOREIGN KEY fk_RicambiRicambiUtilizzati (CodRicambio)
        REFERENCES Ricambi (CodRicambio)
        ON UPDATE CASCADE,
    FOREIGN KEY fk_InterventiManutenzioneRicambiUtilizzati (CodInterManut)
        REFERENCES InterventiManutenzione (CodInterManut)
        ON UPDATE CASCADE
    ) ENGINE = InnoDB;

/*Abilito le Foregin Key*/
SET FOREIGN_KEY_CHECKS = 1;


/*SCRIPT INSERIMENTO RECORD INDISPENSABILI*/

/*Campi di uno stato della richiesta: CodStato, DescrStatoRich*/
INSERT INTO StatiRichieste VALUES ('ATT', 'In Attesa'), /*Stato di default per le Richieste*/
                                  ('ANN', 'Annullata'),
                                  ('RIF', 'Rifiutata'),
                                  ('ESE', 'Eseguita'),
                                  ('INE', 'In Esecuzione');
```

QUERY, PROCEDURE, TRIGGER E FUNZIONI

QUERY/VISTE

Query1 - Vista

Di ogni richiesta eseguita estrarre: codice, data generazione, l'operatore che l'ha effettuata, la data in cui è stata generata, la data in cui è stata evasa, la descrizione dello stato attuale, il tempo in giorni trascorso da quando è stata generata a quando è stata accettata (TempoPendenza), in quanti giorni è stata ultimata dopo essere stata accettata (TempoEvasione), quanti giorni sono trascorsi da quando è stata generata a quando è stata completata (TempoTotale).

```
CREATE VIEW TempoRichieste AS
SELECT rm.IdRichiesta, CONCAT(d.Nome, ' ', d.Cognome) AS Operatore, rm.DataGenerazione,
       rm.DataEvasione, sr.DescrStatoRich, DATEDIFF(MIN(im.DataInizio),
       rm.DataGenerazione) AS TempoPendenza, DATEDIFF(rm.DataEvasione,
       MIN(im.DataInizio)) AS TempoEvasione, DATEDIFF(rm.DataEvasione,
       rm.DataGenerazione) AS TempoTotale
FROM ((RichiesteManutenzione rm NATURAL JOIN StatiRichieste sr) JOIN
     (Operatori o JOIN Dipendenti d ON (o.IdOperatore = d.IdDipendente)) USING (IdOperatore))
     NATURAL JOIN InterventiManutenzione im
WHERE rm.CodStato = 'ESE'
GROUP BY rm.IdRichiesta, rm.DataGenerazione, d.Nome, d.Cognome, sr.DescrStatoRich,
         rm.DataEvasione, rm.DataGenerazione
```

Funzioni MYSQL utilizzate:

- CONCAT(stringa1, stringa2, …, stringaN)
  Dà in uscita il risultato della concatenazione delle stringhe da ″stringa1″ a ″stringaN″
- DATEDIFF(data1, data2)
  Restituisce il risultato della differenza tra data1 e data2 in numero di giorni

Output:

```
+-------------+--------------+---------------------+---------------------+---------------+---------------+---------------+-------------+
| IdRichiesta | Operatore    | DataGenerazione     | DataEvasione        | DescrStatoRich | TempoPendenza | TempoEvasione | TempoTotale |
+-------------+--------------+---------------------+---------------------+---------------+---------------+---------------+-------------+
|           1 | Ash Ketchup  | 2011-06-13 10:11:19 | 2011-06-17 15:17:06 | Eseguita      |             2 |             2 |           4 |
|           5 | Vito Scaletta | 2010-04-04 00:00:00 | 2010-04-09 00:00:00 | Eseguita      |             1 |             4 |           5 |
+-------------+--------------+---------------------+---------------------+---------------+---------------+---------------+-------------+
2 rows in set (0.00 sec)
```

Query2

Trovare la descrizione delle macchine con il massimo numero di richieste di manutenzioni accettate, e riportare tale numero.

```
/*Funzionamento della query:
- le richieste accettate vengono suddivise per macchina (GROUP BY)
- ricavo il numero di richieste accettate per ogni macchina (SOTTOSELECT)
- viene scelto la macchina per cui il numero di richieste accettate è maggiore o uguale del numero
di richieste accettate per ogni macchina (quindi il massimo numero di richieste accettate) (HAVING)
- possono esserci più macchine con lo stesso numero massimo di richieste accettate
*/

SELECT m.DescrMacchina, COUNT(rm.IdRichiesta) AS NumeroRichieste
FROM Macchine m NATURAL JOIN RichiesteManutenzione rm
WHERE rm.CodStato IN ('ESE', 'INE') /*Solo richieste accettate*/
GROUP BY m.CodMacchina
/*Vincolo il count alla PK così da non aver problemi con valori NULL*/
HAVING COUNT(rm.IdRichiesta) >= ALL(
            /*La sottoselect restituisce il numero di richieste accettate per ogni macchina*/
            SELECT Count(*)
            FROM Macchine m NATURAL JOIN RichiesteManutenzione rm
            WHERE rm.CodStato IN ('ESE', 'INE')
            GROUP BY m.CodMacchina)
ORDER BY m.CodMacchina ASC;
```

Output:

```
+--------------------+-----------------+
| DescrMacchina      | NumeroRichieste |
+--------------------+-----------------+
| Macchina del Tempo |               2 |
+--------------------+-----------------+
1 row in set (0.00 sec)
```

Query3 - Vista

Trovare la descrizione dei reparti con il massimo numero di richieste di manutenzioni accettate, e riportare tale numero.

```
/*Non potendo utilizzare una funzione di aggregazione sul risultato di una funzione di aggregazione,
passo attraverso una vista*/

DROP VIEW IF EXISTS RichiestePerReparto;

CREATE VIEW RichiestePerReparto AS
/*Numero di richieste accettate in ogni reparto*/
SELECT r.DescrReparto, COUNT(*) AS NumRichieste
FROM RichiesteManutenzione rm NATURAL JOIN (Macchine m NATURAL JOIN Reparti r)
WHERE rm.CodStato IN ('ESE', 'INE') /*Solo richieste accettate*/
GROUP BY r.DescrReparto;

SELECT DescrReparto, NumRichieste
FROM RichiestePerReparto
WHERE NumRichieste = (SELECT MAX(NumRichieste) FROM RichiestePerReparto)
```

Output:

```
+--------------------+--------------+
| DescrReparto       | NumRichieste |
+--------------------+--------------+
| Ricerca e sviluppo |            3 |
+--------------------+--------------+
1 row in set (0.00 sec)
```

Query4 - Viste

Di ogni richiesta eseguita estrarre: il codice identificativo, il costo degli operatori, il costo dei ricambi, il costo totale.

```
DROP VIEW IF EXISTS CostoManutenzioni;
DROP VIEW IF EXISTS CostoRicambi;
DROP VIEW IF EXISTS CostoManutentori;
DROP VIEW IF EXISTS CostoTotaleRicambi;
DROP VIEW IF EXISTS CostoTotaleManutentori;

CREATE VIEW CostoRicambi AS
/*Calcolo il costo di ogni ricambio utilizzato in interventi relativi ad una richiesta*/
SELECT rm.IdRichiesta,
    CONVERT(
        /*Restituisce il costo dei ricambi utilizzati per gli interventi relativi ad una richiesta*/
        IF((ru.NumeroPezzi * ri.CostoPezzo) IS NOT NULL, ru.NumeroPezzi * ri.CostoPezzo, 0),
        DECIMAL(10, 2)
        ) AS CostoRicambio
FROM RichiesteManutenzione rm
     NATURAL JOIN (InterventiManutenzione im NATURAL LEFT JOIN
     (RicambiUtilizzati ru NATURAL JOIN Ricambi ri)) /*non in tutti gli interventi vengono
utilizzati ricambi*/
WHERE rm.CodStato = 'ESE';


CREATE VIEW CostoManutentori AS
/*Calcolo il costo di ogni operatore che ha generato interventi relativi ad una richiesta*/
SELECT rm.IdRichiesta,
    /*Sommo il munero di ore per il costo orario ed il numero di minuti per il costo al
minuto(calcolato)*/
    CONVERT(
        HOUR(TIMEDIFF(im.DataFine,im.DataInizio)) * d.RetribuzioneOraria +
        MINUTE(TIMEDIFF(im.DataFine,im.DataInizio)) * (d.RetribuzioneOraria / 60)
        , DECIMAL(10, 2)
        ) AS CostoManutentore
FROM (RichiesteManutenzione rm
     NATURAL JOIN InterventiManutenzione im)
     NATURAL JOIN
     (Manutentori m JOIN Dipendenti d ON (m.IdManutentore = d.IdDipendente))
WHERE rm.CodStato = 'ESE';

/*Ho bisogno di queste altre due viste in quanto potrei avere un numero diverso di ricambi ed
operatori che hanno lavorato.
Nel caso appena citato, andrei a fare delle somme errate se lavoro sull'aggregazione secondo
IdRichiesta della giunzione data dalle due viste precedenti*/
CREATE VIEW CostoTotaleRicambi AS
SELECT IdRichiesta, SUM(CostoRicambio) AS CostoTotaleRicambi
FROM CostoRicambi
GROUP BY IdRichiesta;

CREATE VIEW CostoTotaleManutentori AS
SELECT IdRichiesta, SUM(CostoManutentore) AS CostoTotaleManutentori
FROM CostoManutentori
GROUP BY IdRichiesta;

/*Ricavo i costi di ogni richiesta ordinati dalla più costosa alla meno costosa*/
SELECT IdRichiesta, co.CostoTotaleManutentori, cr.CostoTotaleRicambi,
       CONVERT(co.CostoTotaleManutentori + cr.CostoTotaleRicambi, DECIMAL(10, 2)) AS CostoTotale
FROM CostoTotaleManutentori co NATURAL LEFT JOIN CostoTotaleRicambi cr /*Non necessariamente in
unintervento ho utilizzato dei ricambi*/
ORDER BY CostoTotale DESC;
```

Funzioni MySQL utilizzate:

- CONVERT(valore, tipo):
  Converte il valore passato al tipo indicato
- DECIMAL(M, D)
  Tipo di valore numerico di cui vengono specificati: il numero massimo di cifre a destra della virgola(M), il numero massimo di cifre a sinistra della virgola(D)
- TIMEDIFF(data1, data2)
  Restituisce il risultato della differenza tra data1 e data2 come valore DATETIME

- HOUR(data)
  Estrae il numero il ore da un DATETIME
- MINUTE(data)
  Estrae il numero il minuti da un DATETIME
- IF(condizione, valore-Vero, valore-Falso)
  Se la condizone risulta essere vera viene restituito il valore "valore-Vero" altrimenti restituisce "valore-Falso"

Output:

```
+-------------+------------------------+--------------------+-------------+
| IdRichiesta | CostoTotaleManutentori | CostoTotaleRicambi | CostoTotale |
+-------------+------------------------+--------------------+-------------+
|           5 |                4800.00 |              21.00 |     4821.00 |
|           1 |                2400.00 |               3.00 |     2403.00 |
+-------------+------------------------+--------------------+-------------+
2 rows in set (0.00 sec)
```

Query5

Dato un idDipendente, trovare il ruolo del dipendente associato a tale id.

```
/*Sfrutto il fatto che un dipendente può avere un solo ruolo, cerco il suo id tra le tabelle dei
vari tipi di dipendente e faccio l'unione dei risultati*/

SELECT 'M' AS Ruolo
FROM Manutentori
WHERE IdManutentore = $idDipendente

UNION

SELECT 'O' AS Ruolo
FROM Operatori
WHERE IdOperatore = $idDipendente

UNION

SELECT 'S' AS Ruolo
FROM Segretari
WHERE IdSegretario = $idDipendente
```

Output:

con $idDipendente = 4

```
+--------+
| Ruolo |
+--------+
| O     |
+--------+
1 row in set (0.00 sec)
```

Query6

Trovare l'id delle richieste che:

- sono state generate tra il 3 Marzo 2010 ed il 1 Maggio 2010
- sono state generate dopo l'8 Aprile 2010
- contengano la parola "dita" nella descrizione
- siano almeno stati utilizzati i ricambi "Scatola 100 bulloni" e "Tanica da 5l di olio" per riparare la macchina
- siano riferite alla macchina "Caffettiera"
- il reparto sia "Direzione"
- siano state Eseguite

```
SELECT DISTINCT rm.IdRichiesta
FROM ((RichiesteManutenzione rm NATURAL JOIN Macchine m)
      /*Serve a ricavare il CodReparto da Macchine*/
     NATURAL LEFT JOIN InterventiManutenzione im)
     /*Non necessariamente una richiesta ha interventi associati*/
     NATURAL LEFT JOIN RicambiUtilizzati ru
     /*Non necessariamente in un intervento sono stati utilizzati dei ricambi*/
WHERE TRUE /*Serve da riempitivo nel caso non vengano specificate condizioni*/
AND rm.DataGenerazione BETWEEN '2010-3-3' AND '2010-5-1'
AND rm.DataEvasione >= '2010-4-8'
AND rm.DescrRichiesta LIKE '%dita%'
/*Cerca una descrizione della richiesta che contenga la parola dita*/
AND rm.IdRichiesta IN (
    /*Subquery utilizzata per trovare record sulla relazione molti a molti.
     Faccio una JOIN implicita tra due tabelle uguali, usando la virgola nel FROM e le condizioni
     nel WHERE*/
    SELECT DISTINCT rm1.IdRichiesta
    FROM (RichiesteManutenzione rm1 NATURAL JOIN InterventiManutenzione im1)
         NATURAL JOIN RicambiUtilizzati ru1,
         (RichiesteManutenzione rm2 NATURAL JOIN InterventiManutenzione im2)
         NATURAL JOIN RicambiUtilizzati ru2
    WHERE TRUE
        AND rm1.IdRichiesta = rm2.IdRichiesta /*Condizione di giunzione*/
        AND ru1.CodRicambio != ru2.CodRicambio
        /*Controllo che tutti i ricambi siano diversi tra di loro (alta complessità)*/

        /*Controllo che i ricambi siano quelli richiesti*/
        AND ru1.CodRicambio IN (4, 7)
        AND ru2.CodRicambio IN (4, 7))
AND rm.CodMacchina = 8
AND m.CodReparto = 4
AND rm.CodStato = 'ESE'
```

Output:

```
+-------------+
| IdRichiesta |
+-------------+
|           5 |
+-------------+
1 row in set (0.00 sec)
```

Query7

Trovare la descrizione delle macchine su cui non è mai stata generata una richiesta di manutenzione.

```
/*Una macchina non ha richiesta di manutenzione associate se il suo codice non compare in nessuna
delle richieste nel sistema*/
SELECT DescrMacchina
FROM Macchine m
WHERE NOT EXISTS(
    /*Macchine associate ad almeno una richiesta*/
    SELECT DISTINCT CodMacchina
    FROM RichiesteManutenzione rm
    WHERE m.CodMacchina = rm.CodMacchina)
```

Output:

```
+----------------------+
| DescrMacchina        |
+----------------------+
| Trafila 1            |
| Trafila 2            |
| Nintendo WII         |
| Muro di carton-gesso |
| Muro di mattoni      |
| Muro di farina       |
| Pressa Grande        |
| Pressa Piccola       |
| Pressa Media         |
+----------------------+
9 rows in set (0.00 sec)
```

PROCEDURE

Procedure1 - Errore

Questa procedura serve a generare un errore: scrivere in una Tabella che non esiste.

```
DROP PROCEDURE IF EXISTS Errore;

delimiter $

CREATE PROCEDURE Errore ()
BEGIN
    INSERT INTO NonEsiste VALUES ('');
END; $

delimiter ;
```

Procedure2 - ApriIntervento

Questa è utilizzata per aprire un Nuovo Intervento di Manutenzione, assicurandosi che la richiesta a cui fa riferito non sia già stata chiusa.
La Transazione serve ad evitare che a causa del prerilascio venga riaperta una Richiesta chiusa (se ho concorrenza tra questa procedura ed ChiudiRichiesta).

```
DROP PROCEDURE IF EXISTS ApriIntervento;

delimiter $

CREATE PROCEDURE ApriIntervento (IN Richiesta INT, Manutentore INT)
BEGIN
    DECLARE StatoRichiesta CHAR(3);

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    START TRANSACTION;

    /*Voglio che il valore letto in CodStato rimanga lo stesso per tutta la procedura*/
    SELECT CodStato INTO StatoRichiesta
    FROM RichiesteManutenzione
    WHERE IdRichiesta = Richiesta;

    IF StatoRichiesta IN ('ATT', 'INE') THEN
        INSERT INTO InterventiManutenzione (IdRichiesta, IdManutentore, DataInizio)
            VALUES (Richiesta, Manutentore, NOW());
            IF StatoRichiesta = 'ATT' THEN
                UPDATE RichiesteManutenzione SET CodStato = 'INE' WHERE IdRichiesta = Richiesta;
            END IF;
        COMMIT;
    ELSE
        ROLLBACK;
    END IF;
END; $

delimiter ;
```

Procedure3 - ChiudiRichiesta

Questa è utilizzata per chiudere l'intervento su cui il manutentore stà lavorando e l'intera richiesta, purché non ci siano altri interventi aperti relativi ad essa e non sia stata già chiusa.
La Transazione serve ad evitare che a causa del prerilascio venga chiusa una Richiesta con interventi aperti (se ho concorrenza tra questa procedura ed ApriIntervento).
E' di tipo SERIALIZABLE in quanto vado a leggere mediante COUNT un dato che interessa tutta la tabella, che quindi non deve cambiare (inserimento, eliminazione) durante l'esecuzione della procedura.

```
DROP PROCEDURE IF EXISTS ChiudiRichiesta;

delimiter $

CREATE PROCEDURE ChiudiRichiesta (IN Richiesta INT, InterventoManutenzione INT)
BEGIN
    /*Dichiaro le variabili*/
    DECLARE StatoRichiesta CHAR(3);
    DECLARE InterventiAperti INT;

    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
    START TRANSACTION;

    /*Leggo lo stato della richiesta associata*/
    SELECT CodStato INTO StatoRichiesta
    FROM RichiesteManutenzione
    WHERE IdRichiesta = Richiesta;

    /*Vedo se ci sono Interventi ancora aperti, al di fuori di quello che devo chiudere*/
    SELECT COUNT(*) INTO InterventiAperti
    FROM InterventiManutenzione
    WHERE IdRichiesta = Richiesta AND DataFine IS NULL AND CodInterManut !=  InterventoManutenzione;

    IF StatoRichiesta = 'INE' AND InterventiAperti = 0 THEN
        UPDATE RichiesteManutenzione SET CodStato = 'ESE', DataEvasione = NOW() WHERE IdRichiesta =
Richiesta;
        UPDATE InterventiManutenzione SET DataFine = NOW() WHERE CodInterManut =
InterventoManutenzione;
        COMMIT;
    ELSE
        ROLLBACK;
    END IF;
END; $

delimiter ;
```

Procedure4 - AnnullaRichiesta

Questa è utilizzata per annullare una richiesta, assicurandosi che la richiesta sia ancora in stato di Attesa.
La Transazione serve ad evitare che a causa del prerilascio venga rifutata una Richiesta ormai passata allo stato 'In Esecuzione' (se ho concorrenza tra questa procedura ed ApriIntervento)

```
DROP PROCEDURE IF EXISTS AnnullaRichiesta;

delimiter $

CREATE PROCEDURE AnnullaRichiesta (IN Richiesta INT)
BEGIN
    DECLARE StatoRichiesta CHAR(3);

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    START TRANSACTION;

    SELECT CodStato INTO StatoRichiesta
    FROM RichiesteManutenzione
    WHERE IdRichiesta = Richiesta;

    IF StatoRichiesta = 'ATT' THEN
        UPDATE RichiesteManutenzione SET CodStato = 'ANN', DataEvasione = NOW() WHERE IdRichiesta =
Richiesta;
        COMMIT;
    ELSE
        ROLLBACK;
    END IF;
END; $

delimiter ;
```

Procedure5 - RifiutaRichiesta

Questa è utilizzata per rifiutare una richiesta, assicurandosi che la richiesta sia ancora in stato di Attesa.
La Transazione serve ad evitare che a causa del prerilascio venga rifutata una Richiesta ormai passata allo stato 'In Esecuzione' (se ho concorrenza tra questa procedura ed ApriIntervento)

```
DROP PROCEDURE IF EXISTS RifiutaRichiesta;

delimiter $

CREATE PROCEDURE RifiutaRichiesta (IN Richiesta INT)
BEGIN
    DECLARE StatoRichiesta CHAR(3);

    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    START TRANSACTION;

    SELECT CodStato INTO StatoRichiesta
    FROM RichiesteManutenzione
    WHERE IdRichiesta = Richiesta;

    IF StatoRichiesta = 'ATT' THEN
        UPDATE RichiesteManutenzione SET CodStato = 'RIF', DataEvasione = NOW() WHERE IdRichiesta =
Richiesta;
        COMMIT;
    ELSE
        ROLLBACK;
    END IF;
END; $

delimiter ;
```

Procedure6 - RegistraUtente

Questa procedura viene chiamata per Registrare un Utente. Con la registrazione ho:

- la creazione del Dipendete
- la creazione dell'Utente (di default non attivo)
- la creazione del record relativo alla mansione specificata

```
DROP PROCEDURE IF EXISTS RegistraUtente;

delimiter $

CREATE PROCEDURE RegistraUtente (UNome VARCHAR(30), UCognome VARCHAR(30),
                                 Reparto INT, DataAssunz DATE, UCF CHAR(16),
                                 UUsername VARCHAR(15), UPasswordHash CHAR(40),
                                 USesso CHAR(1), Mansione VARCHAR(30))
BEGIN
    DECLARE IdUtente INT;

    INSERT INTO Dipendenti (Nome, Cognome, CodReparto, DataAssunzione, CF, Sesso) VALUES (UNome,
UCognome, Reparto, DataAssunz, UCF, USesso);
    /*Trovo l'id assegnato automaticamente al Dipendete e lo salvo*/
    SET IdUtente = LAST_INSERT_ID();
    INSERT INTO Utenti(Username, PasswordHash, IdDipendente) VALUES  (UUsername, UPasswordHash,
IdUtente);
    /*inserisco l'utente in base alla mansione specificata*/
    CASE Mansione
        WHEN 'manutentore' THEN INSERT INTO Manutentori (IdManutentore) VALUES (IdUtente);
        WHEN 'operatore' THEN INSERT INTO Operatori (IdOperatore) VALUES (IdUtente);
        WHEN 'segretario' THEN INSERT INTO Segretari (IdSegretario) VALUES (IdUtente);
    END CASE;
END; $

delimiter ;
```

Nota: La funzione LAST_INSERT_ID() di MySQL restituisce l'ultimo valore AUTO_INCREMENT inserito. Il DBMS, secondo le specifiche della funzione, garantisce il corretto funzionamento

anche con accessi concorrenti. Una Transazione non è quindi necessaria.

TRIGGER

Trigger1 - DateIns_DoppiInteventi & DateAgg

Sono elencati due trigger in quanto entrambi fanno un controllo sulle stesse date, uno all'inserimento, uno all'aggiornamento.
Il primo però contiene altri controllo esplicitati qui sotto.

Questo trigger serve ad assicurare, quando viene INSERITO un intervento:

- un manutentore possa avere un solo intervento aperto per volta
- la data di fine intervento sia più grande di quella d'inizio

```
DROP TRIGGER IF EXISTS DateIns_DoppiInteventi;

delimiter $

CREATE TRIGGER DateIns_DoppiInteventi
BEFORE INSERT ON InterventiManutenzione
FOR EACH ROW
BEGIN
    DECLARE InterventiManutentore INT;

    /*Conto gli interventi aperti per il manutentore considerato*/
    SELECT COUNT(*) INTO InterventiManutentore
    FROM InterventiManutenzione
    WHERE IdManutentore = NEW.IdManutentore AND DataFine IS NULL;

    /*Se ha già un intervento aperto, non può aprirne un altro*/
    IF (InterventiManutentore > 0)
        THEN CALL Errore;
    END IF;

    /*Se la data d'inizio è maggiore di quella di fine, c'è un errore*/
    IF (NEW.DataInizio > NEW.DataFine)
        THEN CALL Errore;
    END IF;
END; $

delimiter ;
```

Questo trigger serve ad assicurare, quando viene AGGIORNATO un intervento, la data di fine intervento sia più grande di quella d'inizio

```
DROP TRIGGER IF EXISTS DateAgg;

delimiter $

CREATE TRIGGER DateAgg
BEFORE UPDATE ON InterventiManutenzione
FOR EACH ROW
BEGIN
    IF (NEW.DataInizio > NEW.DataFine)
        THEN CALL Errore;
    END IF;
END; $

delimiter ;
```

Trigger2 - Retribuzione

Assicura che retribuzione oraria di un dipendete resti maggiore o uguale a zero

```
DROP TRIGGER IF EXISTS Retribuzione;
```

```
delimiter $

CREATE TRIGGER Retribuzione
BEFORE UPDATE ON Dipendenti
FOR EACH ROW
BEGIN
    IF (NEW.RetribuzioneOraria < 0)
        THEN CALL Errore;
    END IF;
END; $

delimiter ;
```

Trigger3 - RichiestePerMacchina

Assicura che vi sia una sola richiesta aperta per macchina.
Per aprirne un'altra quella aperta deve essere chiusa.
Potrebbe servire una transazione serializable per evitare che due operatori generino contemporaneamente una richiesta sulla stessa macchina, ma MySQL non lo permette.

```
DROP TRIGGER IF EXISTS RichiestePerMacchina;

delimiter $

CREATE TRIGGER RichiestePerMacchina
BEFORE INSERT ON RichiesteManutenzione
FOR EACH ROW
BEGIN
    DECLARE NumRichieste INT;

    /*Conto le richieste aperte per la macchina considerata*/
    SELECT COUNT(*) INTO NumRichieste
    FROM RichiesteManutenzione
    WHERE CodMacchina = NEW.CodMacchina AND DataEvasione IS NULL;

    /*Non deve essercene nessuna*/
    IF (NumRichieste > 0)
        THEN CALL Errore;
    END IF;
END; $

delimiter ;
```

INTERFACCIA WEB - DESCRIZIONE

L'interfaccia web creta è stata pensata per un uso da parte di diverse tipologie di utenti.
Ogni utente prima di poter accedere al sistema deve registrarsi e la sua registrazione deve essere accettata.
Gli utenti registrati condividono:

- una pagina di login comune, dalla quale vengono poi indirizzati a sezioni diverse del sito.
- una pagina che permette di cambiare la propria password ed il numero di righe visualizzate su alcune tabelle presenti nelle varie pagine web (impostazione uguale dentro alle diverse sezioni dei vari utenti)
- il collegamento ad una pagina che permette di fare il logout, solo ad utenti che hanno fatto il login

Gli utenti possono essere:

- Operatori, i quali possono:
    - Generare richieste di manutenzione relative alle macchine del reparto in cui si trovano

    - Annullare richieste generate
    - Vedere l'evolversi di una richiesta effettuata
    - Visualizzare lo storico di tutte le richieste da loro generate
- Manutentori, essi possono:
    - Visualizzare le richieste relative alle macchine
    - Scegliere se rifiutare richieste in Attesa
    - Accettare una richiesta, quindi generare un Intervento di Manutenzione, ed a livello pratico, iniziare a lavorare su di una macchina
    - Utilizzare un numero indefinito di ricambi durante il lavoro
    - Quando il lavoro è terminato possono:
        - Chiudere l'intervento:
          la macchina non è ancora operativa, ma il manutentore ha ultimato le riparazioni del suo ambito di lavoro o è giunto a fine turno.
        - Chiudere la richiesta:
          ciò implica che la macchina risulta essere pienamente operativa.
    - Visualizzare lo storico delle richieste
    - Visualizzare lo storico degli interventi
- Segretari, tali dipendenti sono abilitati a:
    - Accettare utenti che fanno richiesta di accedere al sistema
    - Inserire o Eliminare: Reparti, Macchine, Utenti
    - Ricercare Richieste imponendo parametri precisi (es: l'oratore che l'ha generata, il periodo di generazione, i manutentori che hanno lavorato per riporatare all'operatività la macchina, ecc..)
    - Visualizzare i costi delle Richieste Eseguite
    - Visualizzare i tempi relativi alle Richieste
    - Visualizzare le macchine su cui sono state generate più Richieste
    - Visualizzare le reparto su cui sono state generate più Richieste

Autenticazione e Mantenimento dello Stato

Si è scelto di salvare il nome utente e l'hash della password all'interno del database.
Tali dati vengono scritti alla registrazione e letti al login.
Se l'autenticazione va a buon fine, l'identificativo dell'utente e la sua mansione vengono salvati su variabili di sessione, utilizzate come strumento per mantenere lo stato tra pagine.
All'apertura di una pagina, tali variabili vengono lette per verificare che l'utente sia in una sezione in cui è abilitato ad accedere.
Vi è inoltre una pagina accendibile da qualsiasi utente, sia loggato che non loggato: credits.php.
Anche un cookie è stato utilizzato per il mantenimento dello stato tra pagine, in cui viene ricordata la pagina precedentemente visitata per permettere di tornare indietro durante la navigazione.

Accedere al sito

E' possibile accedere al sito dalla pagina:
http://basidati/basidati/~fcesarat/index.php