|  |  |
| --- | --- |
|  |  |

Basi di Dati e Conoscenza

Progetto A.A. 2018/2019

TITOLO DEL PROGETTO

Matricola

Nome e Cognome

**Indice**

[1. Descrizione del Minimondo 2](#_Toc606296459)

[2. Analisi dei Requisiti 5](#_Toc1289394997)

[3. Progettazione concettuale 9](#_Toc2081466291)

[4. Progettazione logica 11](#_Toc2147004904)

[5. Progettazione fisica 16](#_Toc518560220)

[Appendice: Implementazione 18](#_Toc403811585)

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | Realizzare un servizio di “directory aziendale”, che consenta di tenere traccia di tutte le  informazioni legate ai recapiti ed alle mansioni di un’azienda.  Ogni dipendente dell’azienda, identificato da codice fiscale, nome, cognome, data di nascita,  luogo di nascita, indirizzo di residenza svolge una particolare mansione all’interno dell’azienda.  Tali mansioni vengono svolte in differenti uffici dell’azienda. L’azienda ha a disposizione un  numero arbitrario di edifici, ciascuno di un numero arbitrario di piani. In ciascun piano può  esserci un numero arbitrario di uffici. Ciascun ufficio è assegnato ad una mansione specifica  dell’azienda. Un ufficio ha a disposizione un certo numero di postazioni, ciascuno associato ad  un numero telefonico interno ed esterno. La gestione degli uffici, piani, edifici è demandata al  Settore Spazi dell’azienda, anch’esso composto da dipendenti assegnati a particolari uffici.  L’assegnazione di un dipendente ad un ufficio avviene su base di turnazioni periodiche. Ogni  dipendente, pertanto, è associato alla data dell’ultimo trasferimento. [Periodicamente, il sistema  indica qual è l’insieme dei dipendenti che deve essere soggetto a trasferimento.] Su base  periodica, l’ufficio spazi genera un report indicante tutti i dipendenti, raggruppati per mansione,  che devono essere spostati.  Un dipendente dell’ufficio spazi deve poter effettuare un trasferimento di dipendente.  Taletrasferimento deve essere effettuato rispettando l’associazione tra le mansioni e gli uffici.  Pertanto, dato un dipendente da trasferire, il dipendente del Settore Spazi può effettuare uno  scambio tra due dipendenti assegnati alla stessa mansione, o individuare una postazione libera  utilizzabile per quella mansione. Si noti che un dipendente non può essere riassegnato ad una  postazione in cui era già stato assegnato nei passati tre anni.  Il Settore Amministrativo può decidere in qualsiasi momento di modificare la mansione di un  dipendente. In quel caso, il Settore Spazi troverà tale dipendente indicato come da trasferire.  Inoltre, il settore amministrativo può generare, per ciascun dipendente, un report indicante a  quali uffici esso è stato assegnato nel tempo.  Ciascun dipendente ha accesso alla directory aziendale per conoscere l’attuale ubicazione di un  qualsiasi altro dipendente, ricercandolo per nome, per cognome o utilizzando entrambi. Inoltre,  un dipendente può ricercare un certo numero di telefono interno per sapere a quale ufficio,  piano, edificio e mansione questo è associato, scoprendo anche quale dipendente è attualmente  associato allo stesso e sapendo se esso è in procinto di essere trasferito o meno  Si noti che, quando si effettua la ricerca di un dipendente, è necessario restituire tutti i recapiti  associati allo stesso. In particolare, un dipendente ha un indirizzo email personale ed uno  associato all’ufficio a cui fa capo. |

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

|  |  |  |  |
| --- | --- | --- | --- |
| **Linea** | **Termine** | **Nuovo termine** | **Motivo correzione** |
| 2 | Di un’azienda | Dei dipendenti di un’azienda | Le informazioni delle quali si vuole tenere traccia sono relative ai dipendenti che lavorano all’interno dell’azienda |
| 5 | Tali mansione | Ogni mansione |  |
| 8 | Ciascuno | Ed è | Ad ogni ufficio è associate un numero telefonico interno ed esterno, non alla singola postazione |
| 12 | Data dell’ultimo trasferimento | Storico delle assegnazioni | Per rispettare il vincolo temporale della riassegnazione ad una postazione, devo mantenere non solo l’ultimo trasferimento, ma tutti quanti |
| 28 | Numero di telefono | Numero di telefono interno |  |
| 29 | Quale dipendente | Quali dipendenti | All’interno di un ufficio sono presenti più di un dipendente, non un singolo |

### Specifica disambiguata

|  |
| --- |
| Realizzare un servizio di “directory aziendale”, che consenta di tenere traccia di tutte le informazioni legate ai recapiti ed alle mansioni dei dipendenti di un’azienda.  Ogni dipendente dell’azienda, identificato da codice fiscale, nome, cognome, data di nascita, luogo di nascita, indirizzo di residenza svolge una particolare mansione all’interno dell’azienda. Ogni mansione viene svolta in differenti uffici dell’azienda. L’azienda ha a disposizione un numero arbitrario di edifici, ciascuno di un numero arbitrario di piani. In ciascun piano può esserci un numero arbitrario di uffici. Ciascun ufficio è assegnato ad una mansione specifica dell’azienda. Un ufficio ha a disposizione un certo numero di postazioni, ed è associato ad un numero telefonico interno ed esterno. La gestione degli uffici, piani, edifici è demandata al Settore Spazi dell’azienda, anch’esso composto da dipendenti assegnati a particolari uffici.  L’assegnazione di un dipendente ad un ufficio avviene su base di turnazioni periodiche. Per ogni dipendente viene mantenuto lo storico delle assegnazioni. Periodicamente, il sistema indica qual è l’insieme dei dipendenti che deve essere soggetto a trasferimento. Su base periodica, l’ufficio spazi genera un report indicante tutti i dipendenti, raggruppati per mansione, che devono essere spostati.  Un dipendente dell’ufficio spazi deve poter effettuare un trasferimento di dipendente. Tale trasferimento deve essere effettuato rispettando l’associazione tra le mansioni e gli uffici. Pertanto, dato un dipendente da trasferire, il dipendente del Settore Spazi può effettuare uno scambio tra due dipendenti assegnati alla stessa mansione, o individuare una postazione libera utilizzabile per quella mansione. Si noti che un dipendente non può essere riassegnato ad una postazione in cui era già stato assegnato nei passati tre anni.  Il Settore Amministrativo può decidere in qualsiasi momento di modificare la mansione di un dipendente. In quel caso, il Settore Spazi troverà tale dipendente indicato come da trasferire. Inoltre, il settore amministrativo può generare, per ciascun dipendente, un report indicante a quali uffici esso è stato assegnato nel tempo.  Ciascun dipendente ha accesso alla directory aziendale per conoscere l’attuale di un qualsiasi altro dipendente, ricercandolo per nome, per cognome o utilizzando entrambi. Inoltre, un dipendente può ricercare un certo numero di telefono interno per sapere a quale ufficio, piano, edificio e mansione questo è associato, scoprendo anche quali dipendenti sono attualmente associati allo stesso e sapendo se, e chi, è in procinto di essere trasferito.  Si noti che, quando si effettua la ricerca di un dipendente, è necessario restituire tutti i recapiti associati allo stesso. In particolare, un dipendente ha un indirizzo email personale ed uno associato all’ufficio a cui fa capo. |

## Glossario dei Termini

|  |  |  |  |
| --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Dipendente | Una persona che lavora all’interno dell’azienda |  | Postazione, Ufficio, Mansione |
| Mansione | Tipo di lavoro svolto all’interno dell’azienda |  | Ufficio, Dipendente |
| Postazione | Postazione di lavoro del singolo dipendente |  |  |
| Ufficio | Luogo fisico dove sono presenti diverse postazioni che svolgono la stessa mansione | Ubicazione | Postazione, Mansione, Dipendente, |
| Assegnazione | Operazione che associa un dipendente ad una postazione per lo svolgimento di una particolare mansion | Trasferimento | Dipendente, Postazione, Ufficio |

## Raggruppamento dei requisiti in insiemi omogenei

|  |
| --- |
| **Frasi relative a ... Dipendente** |
| * Ogni dipendente dell’azienda, identificato da codice fiscale, nome, cognome, data di nascita, luogo di nascita, indirizzo di residenza svolge una particolare mansione all’interno dell’azienda. * Ogni dipendente svolge una particolare mansione * Un Dipendente può essere assegnato al Settore spazi, Settore amministrativo o ad un settore che svolge una qualsiasi altra mansione * Per ogni dipendente viene mantenuto lo storico delle assegnazioni. * Un dipendente dell’ufficio spazi deve poter effettuare un trasferimento di dipendente * Dato un dipendente da trasferire, il dipendente del Settore Spazi può effettuare uno scambio tra due dipendenti assegnati alla stessa mansione, o individuare una postazione libera all’interno di un ufficio che svolge la stessa mansione. * Un dipendente non può essere riassegnato ad una postazione in cui era già stato assegnato nei passati tre anni * Ciascun dipendente ha accesso alla directory aziendale per conoscere l’attuale ubicazione di un qualsiasi altro dipendente * Un dipendente può ricercare un certo numero di telefono interno per sapere a quale ufficio, piano, edificio e mansione questo è associato * A ciascun dipendente è associato un indirizzo email personale ed uno relativo all’ufficio di appartenenza * Quando si effettua la ricerca di un dipendente, è necessario restituire tutti i recapiti associati allo stesso |

|  |
| --- |
| **Frasi relative a ... Mansione** |
| * Ogni mansione è svolta da un numero diversi di dipendenti * Ogni mansione viene svolta in differenti uffici dell’azienda * La mansione di un dipendente può essere modificata da un dipendente del Settore Amministrativo |

|  |
| --- |
| **Frasi relative a ... Ufficio** |
| * Un ufficio ha a disposizione un certo numero di postazioni * A ciascun ufficio è associato un numero telefonico interno ed esterno. * A ciascun ufficio è associato un indirizzo email |

|  |
| --- |
| **Frasi relative a ... Postazione** |
| * Ogni postazione è associata ad un ufficio * Ad ogni postazione è assegnato un dipendente che svolge una particolare mansione |

|  |
| --- |
| **Frasi relative a ... Assegnazione** |
| * L’assegnazione di un dipendente ad una postazione avviene su base di turnazioni periodiche * Un dipendente non può essere riassegnato ad una postazione in cui era già stato assegnato nei passati tre anni. |

# Progettazione concettuale

## Costruzione dello schema E-R

### Integrazione finale

## Regole aziendali

* Un dipendente non deve essere assegnato ad una postazione se il tempo trascorso dall’ultima assegnazione alla stessa postazione è inferiore ai tre anni
* L’assegnazione dei dipendenti deve avvenire solamente alle postazioni attive
* Un dipendente deve essere assegnato ad una postazione con la sua stessa mansione
* I dipendenti dei settori spazi ed amministrativo devono essere associati alle relative mansioni
* **Numero Postazioni Libere** si ottiene tramite l’associazione Composizione
* **Un Ufficio** deve decrementare il *numero di Postazioni* libere dopo l’assegnazione di un dipendente ad una sua postazione attiva precedentemente libera
* **Indirizzo E-Mail ufficio (Dipendente)** si ottiene da **Ufficio** tramite le associazioni **Impiego corrente**, **Assegnazione** e **Composizione**
* Un **Dipendente** deve impostare l’attributo **Da Trasferire** a **True** dopo che la sua mansione è stata modificata
* Una **Postazione attiva** deve impostare l’attributoDisponibile a **False** dopo l’assegnazione di un dipendente ad essa
* Dopo l’assegnazione ad una nuova postazione, l’impiego corrente deve essere inserito tra gli impieghi passati

## Dizionario dei dati

|  |  |  |  |
| --- | --- | --- | --- |
| **Entità** | **Descrizione** | **Attributi** | **Identificatori** |
| Dipendente |  | Nome, Cognome, Data di nascita, Luogo di nascita, Indirizzo residenza, Indirizzo e-mail, Indirizzo e-mail ufficio, Da Trasferire | Codice Fiscale |
| Dipendente Settore Amministrativo |  | Eredita da: Dipendente | Eredita da: Dipendente |
| Dipendete Settore Spazi |  | Eredita da: Dipendente | Eredita da: Dipendente |
| Altro Dipendente |  | Eredita da: Dipendente | Eredita da: Dipendente |
| Impiego |  | Inizio | ID, Assegnazione (Postazione), Matricola (Dipendente) |
| Impiego corrente |  | Eredita da: Impiego | Eredita da: Impiego |
| Impiego passato |  | Eredita da: Impiego, Fine | Eredita da: Impiego |
| Postazione |  |  | ID, Ufficio (Ufficio) |
| Postazione attiva |  | Disponibile | Eredita da: Postazione |
| Ufficio |  | Telefono interno, Telefono esterno, E-mail, Piano, Edificio, Numero postazioni libere | ID |
| Mansione |  | Nome mansione | ID |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| **Concetto nello schema** | **Tipo[[1]](#footnote-2)** | **Volume atteso** |
| D-I | R | 15 \* 2000 |
| Assegnazione | R | 3 \* 2000 |
| Composizione | R | 2100 |
| Svolgimento | R | 105 |
| Occupazione | R | 2000 |
| Dipendente | E | 2100 |
| Dipendente Settore Spazi | E | 100 |
| Dipendente Settore Amministrativo | E | 1900 |
| Altro Dipendente | E | 50 |
| Impiego | E | 15 \* 2000 |
| Impiego Attuale | E | 2000 |
| Impiego passato | E | 14 \* 2000 |
| Postazione | E | 2100 |
| Postazione attiva | E | 1800 |
| Ufficio | E | 110 |
| Mansione | E | 10 |
| Impiego attuale | R | 2000 |

## Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| **Cod.** | **Descrizione** | **Frequenza attesa** |
| 1 | Generazione Report Utenti da trasferire |  |
| 2 | Trasferimento Dipendente (Altro dipendente) |  |
| 3 | Trasferimento Dipendente (Postazione Libera) |  |
| 4 | Modifica Mansione Dipendente |  |
| 5 | Generazione Report Assegnazioni Passate |  |
| 6 | Ricerca Altro dipendente |  |
| 7 | Ricerca ufficio |  |
| 8 | Disabilita postazione |  |
| 9 | Abilita potazione |  |
| 10 | Inserisci postazione |  |
| 11 | Elimina postazione |  |

## Costo delle operazioni

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trasferimento Dipendente (Altro dipendente)** | | | | |
| **Concetto nello schema** | **Tipo** | **Tipo di accesso** | **Descrizione** | **N° accessi** |
| Dipendente | E | Lettura | Leggo l’attuale mansione del dipenente 1 | 1 |
| Occupazione | R | Lettura | Leggo la postazione attuale del dipendente 1 | 1 |
| Assegnazione Attuale | R | Lettura | Trovo I dipendenti non associati alla postazione nei precedenti tre anni | 13 |
| Dipendente | E | Lettura | Controllo che il dipendente 2 sia da trasferire | 1 |
| Dipendente | E | Lettura | Leggo l’attuale mansione del dipendente 2 | 1 |
| Assegnazione | R | Lettura | Leggo la postazione attuale del dipendente 2 | 1 |
| Assegnazione | R | Scrittura | Controllo che il dipendente 1 non sia stato associato nei precedenti tre anni all’attuale postazione del dipendente 2 | 13 |
| Impiego | E | Scrittura | Modifico l’impiego corrente dei dipendenti 1 e 2 | 2\*2 |
| Impiego passato | E | Scrittura | Scrivo l’attuale impiego come impiego passato | 2\*1 |
| Impiego Passato | E | Scrittura | Modifico l’assegnazione della postazione attuale dei dipendenti 1 e 2 | 2\*2 |
| Impiego Attuale | E | Scrittura | Modifico gli indirizzi E-Mail relativi all’ufficio dei dipendenti 1 e 2 | 2\*2 |
| Costo operazione: | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trasferimento Dipendente (Postazione Libera)** | | | | |
| **Concetto** | **Tipo** | **Tipo di accesso** | **Descrizione** | **N° accessi** |
| Mansione dipendente | R | Lettura | Leggo l’attuale mansione del dipendente | 1 |
| Svolgimento |  | Lettura | Leggo tutti gli uffici, con postazioni attive disponibili, che svolgono la stessa mansione del dipendente | 11 |
| Composizione |  | Lettura | Leggo tutte le postazioni libere dell’ufficio | 17 |
| Impiego corrente |  | Lettura | Leggo l’impiego corrente del dipendente | 1 |
| Impiego passato |  | Scrittura | Scrivo impiego corrente in impiego passato | 1 |
| Impiego corrente + Del |  | Scrittura | Modifico l’impiego corrente | 2 |
| Impiego attuale + Del |  | Scrittura | Modifico l’assegnazione dell’impiego attuale | 2 |
| Postazione attiva |  | Scrittua | Imposto la postazione come non disponibile | 1 |
| Ufficio |  | Scrittura | Decremento il numero di postazioni attive disponibili nell’ufficio | 1 |
| Dipendente |  | Scrittura | Modifico l’indirizzo e-mail relativo all’ufficio | 1 |
| Costo operazione: | | | | |

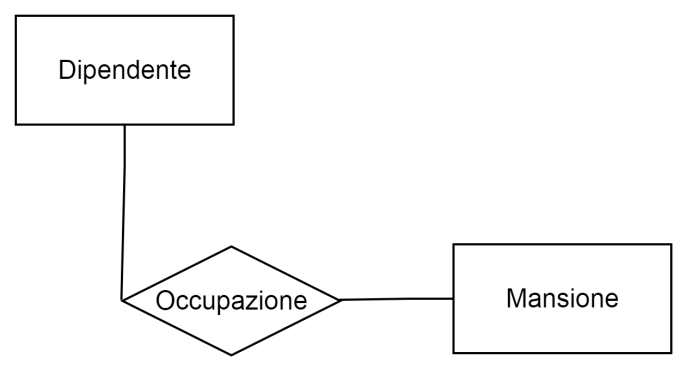
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modifica Mansione Dipendente** | | | | |
| **Concetto** | **Tipo** | **Tipo di accesso** | **Descrizione** | **N° accessi** |
| Mansione dipendente | R | Lettura | Leggo l’attuale mansione del dipendente | 1 |
| Dipendente | E | Scrittura | Imposto il dipendente come da trasferire | 1 |
| Mansione dipendente + Del | R | Scrittura | Scrivo la nuova mansione del dipendente | 1 |
| Costo operazione: | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Abilita postazione** | | | | |
| **Concetto** | **Tipo** | **Tipo di accesso** | **Descrizione** | **N° accessi** |
| Postazione | E | Lettura | Leggo la postazione | 1 |
| Dipendente | E | Scrittura | Inserisco/elimino la postazione la postazione | 1 |
| Costo operazione: | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Inserisci postazione / Elimina postazione** | | | | |
| **Concetto** | **Tipo** | **Tipo di accesso** | **Descrizione** | **N° accessi** |
| Mansione dipendente | R | Scrittura | Inserisco la postazione | 1 |
| Costo operazione: | | | | |

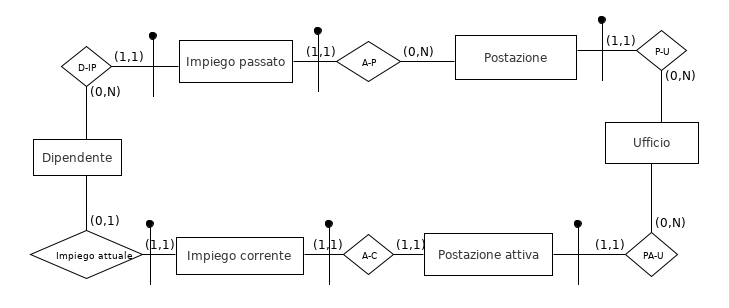
## Ristrutturazione dello schema E-R

**Ridondanza Associazione *Occupazione/ E-Mail ufficio / Postazioni disponibili***

La presenza della relazione **Mansione dipendente** genera ridondanza, ma a fronte di un minimo spreco di memoria, mi permette di accedere direttamente all’informazione relativa al dipendente, senza navigare lo schema per intero. Quella sulla mansione del singolo dipendente è un’informazione ricorrente all’interno delle varie opeazioni.

Altre ridondanze sono gli attributi **E-Mail ufficio** e **Postazioni disponibili**, rispettivamente delle entità *Dipendente* ed *Ufficio*. Ridondanze necessarie per le operazioni, quindi sempre bla, bla, bla...

**Eliminazione Generalizzazioni**

Per permettere una stoicizzazione delle postazioni, oltre agli impieghi, bla, bla, bla, blaaaa...

Per quanto riguarda la generalizzazione per il tipo di Dipendente, visto che tutte e tre le entità condividono gli stessi attribute, aggiungo nell’entità padre due attribute per distinguere il Dipendente dei Settori Spazi e Settore amministrativo dagli altri Dipendenti.

**Identificatori Primari**

|  |  |
| --- | --- |
| **Entità** | **Identificatore Primario** |
| Dipendente | Matricola |
| Impiego Attuale | MatricolaDipendente (Dipendente), IDPostazione (Postazione) |
| Impiego Passato | MatricolaDipendente (Dipendente), IDPostazione (Postazione) |
| Postazione | ID, IDUfficio (Ufficio) |
| Postazione attiva | ID, IDUfficio (Ufficio) |
| Ufficio | ID |
| Mansione | ID |

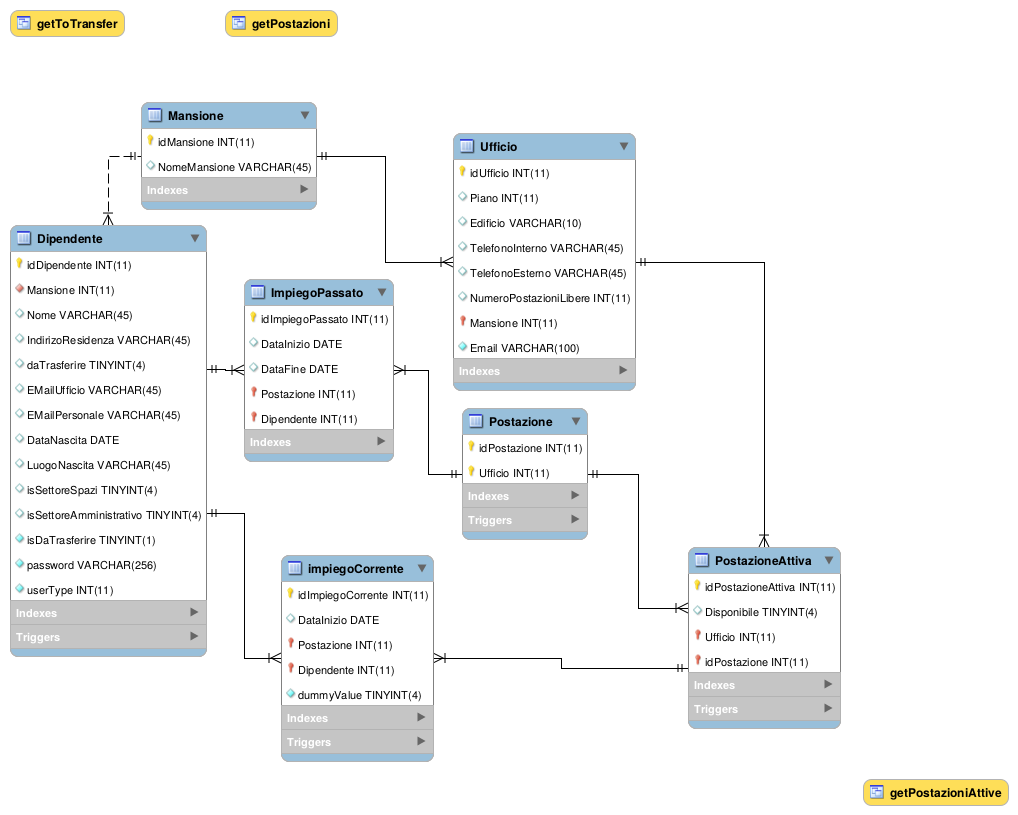
## Trasformazione di attributi e identificatori

|  |  |  |
| --- | --- | --- |
| **Entità** | **Identificatore** | **Nuovo Identificatore** |
| Impiego corrente | Matricola (Dipendente) | ID |
| Impiego passato | Matricola (Dipendente) | ID |
| Postazione | ID (Ufficio) | ID |

## Traduzione di entità e associazioni

|  |  |
| --- | --- |
| **Nome** |  |
| Dipendente | Matricola, Nome, cognome, Indirizzo residenza, Da trasferire, E-Mail personale, E-Mail ufficio, Data di nascita, Luogo di nascita, isSettoreSpazi, isSettoreAmministrativo, **Mansione** |
| Mansione | ID, Nome mansione |
| Ufficio | ID, Piano, Edificio, Telefono interno, Telefono esterno, Numero postazioni libere, **Mansione** |
| Postazione | ID, **Ufficio** |
| Postazione attiva | ID, Disponibile, **Ufficio** |
| Impiego attuale | ID, **Dipendente**, Inizio, **Postazione** |
| Impiego passato | ID, **Dipendente**, Inizio, Fine, **Postazione** |

\***Grassetto** indica chiave esterna, Sottolineato indica chiave primaria



## Normalizzazione del modello relazionale

Effettuare la normalizzazione del modello relazionale precedentemente descritto (in forma grafica) andando a mostrare le forme 1NF, 2NF, 3NF.

2FN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabella<PostazioneAttiva>** | | | | | | |
| **IdUfficio** | **Mansione** | | | **Piano** | | Edificio |
| **1** | 1 | | | 1 | | 1 |
| 2 | | 1 | 1 | | 1 | |
| 3 | | 2 | 2 | | 1 | |
| 4 | | 3 | 2 | | 1 | |

Mansione dipendente da Piano ed Edificio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mansione** | | **Piano** | | Edificio |
| 1 | | 1 | | 1 |
| 1 | 1 | | 1 | |
| 2 | 2 | | 1 | |
| 3 | 2 | | 1 | |

|  |  |  |  |
| --- | --- | --- | --- |
| **IdUfficio** | **Mansione** | | |
| **1** | 1 | | |
| 2 | | 1 |  |
| 3 | | 2 |  |
| 4 | | 3 |  |

Mansione dipendente solamente da Edificio

|  |  |  |  |
| --- | --- | --- | --- |
| **IdUfficio** | **Mansione** | | Piano |
| **1** | 1 | | 1 |
| 2 | | 1 | 1 |
| 3 | | 2 | 1 |
| 4 | | 3 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Mansione** | **Edificio** | | |
| 1 | 1 | | |
| 1 | | 1 |  |
| 2 | | 2 |  |
| 3 | | 2 |  |

# Progettazione fisica

## Utenti e privilegi

|  |  |  |  |
| --- | --- | --- | --- |
| **Utente<tempUser>** | | | |
| **Lettura** | **Aggiornamento** | **Eliminazione** | Inserimento |
| **Dipendente** |  |  |  |
| **Motivazione:** L’utente temUser è necessario per la sola operazione di Login, dunque deve solamente accedere alla tabella Dipendente per controllare la corrispondenza tra il numero del dipendente e la password associata | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Utente<otherUser>** | | | |
| **Lettura** | **Aggiornamento** | **Eliminazione** | Inserimento |
| **Dipendente, ImpiegoPassato, ImpiegoCorrente, Postazione, Postazione Attiva, Mansione** |  |  |  |
| **Motivazione:** Tutti gli altri utenti, che non appartengono al settore amministrativo o al settore spazi, hanno solamente la possibilità di effettuare delle ricerche all’interno della base di dati, senza poter effettuare operazioni di modifica dei dati**.** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Utente<amministrativoUser>** | | | |
| **Lettura** | **Aggiornamento** | **Eliminazione** | Inserimento |
| **Dipendente, ImpiegoPassato, ImpiegoCorrente, Postazione, Postazione Attiva, Mansione** | Dipendente |  |  |
| **Motivazione:** L’utente del settore amministrativo deve poter modificare l’assegnazione del dipendente ad una mansione ed indicare l’utente come da trasferire, oltre ad avere accesso alla ricerca di altri utenti. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Utente<spaziUser>** | | | |
| **Lettura** | **Aggiornamento** | **Eliminazione** | Inserimento |
| **Dipendente, ImpiegoPassato, ImpiegoCorrente, Postazione, Postazione Attiva, Mansione** | Postazione, PostazioneAttiva, Ufficio, ImpiegoCorrente, ImpiegoPassato | Postazione, PostazioneAttiva | Postazione, ImpiegoPassato, PostazioneAttiva |
| **Motivazione:** L’utente del settore spazi, deve poter modificare l’assegnazione dei singoli utenti alle varie postazioni ed aggiungere/rimuovere/modificare le varie postazioni, oltre ad avere accesso alla ricerca di altri utenti. | | | |

## Strutture di memorizzazione

|  |  |  |
| --- | --- | --- |
| **Tabella <ImpiegoPassato>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[2]](#footnote-3)** |
| **IdImpiegoPassato** | Int | NN, AI, PK |
| **Postazione** | Int | NN |
| **Dipendente** | Int | NN |
| **DataInizio** |  |  |
| **DataFine** |  |  |

|  |  |  |
| --- | --- | --- |
| **Tabella <Mansione>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[3]](#footnote-4)** |
| **IdMansione** | Int | NN, AI, PK |
| **NomeMansione** | Int | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella <Postazione>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[4]](#footnote-5)** |
| **IdPostazione** | Int | NN, AI, PK |
| **Ufficio** | Int | NN |

|  |  |  |
| --- | --- | --- |
| **Tabella <PostazioneAttiva>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[5]](#footnote-6)** |
| **IdPostazioneAttiva** | Int | NN, AI, PK |
| **Disponibile** | Int | NN |
| **Ufficio** |  |  |
| **IdPostazione** |  |  |

|  |  |  |
| --- | --- | --- |
| **Tabella <Ufficio>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[6]](#footnote-7)** |
| **IdUfficio** | Int | NN, AI, PK |
| **Piano** | Int | NN |
| **Edificio** |  |  |
| **TelefonoInterno** |  |  |
| **TelefonoEsterno** |  |  |
| **NumeroPostazioni** |  |  |
| **Mansione** |  |  |
| **Email** |  |  |

|  |  |  |
| --- | --- | --- |
| **Tabella <ImpiegoCorrente>** | | |
| **Attributo** | **Tipo di dato** | **Attributi[[7]](#footnote-8)** |
| **IdImpiegoCorrente** | Int | NN, AI, PK |
| **DataInizio** | Int | NN |
| **Postazione** |  |  |
| **Dipendente** |  |  |
| **DummyValue** |  |  |

## Indici

Compilare la seguente tabella, per ciascuna tabella del database in cui sono presenti degli indici. Descrivere le motivazioni che hanno portato alla creazione di un indice.

|  |  |
| --- | --- |
| **Tabella <nome>** | |
| **Indice <nome>** | **Tipo[[8]](#footnote-9):** |
| Colonna 1 | <nome> |

## Trigger

**ImpiegoCorrente (Before Insert):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`impiegoCorrente\_BEFORE\_INSERT` **BEFORE** **INSERT** **ON** `impiegoCorrente` **FOR** **EACH** **ROW**

**BEGIN**

**IF** **NOT** **EXISTS** (**SELECT** 1 **FROM** Dipendente **WHERE** idDipendente = **NEW**.Dipendente **AND** isDaTrasferire **is** **true**) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'L\'ID del dipendente non è valido o non disponibile per il trasferimento!';

**END** **IF**;

**END**

**ImpiegoCorrente (After Insert):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`impiegoCorrente\_AFTER\_INSERT` **AFTER** **INSERT** **ON** `impiegoCorrente` **FOR** **EACH** **ROW**

**BEGIN**

**UPDATE** PostazioneAttiva **set** Disponibile = **false** **WHERE** idPostazioneAttiva = **NEW**.Postazione;

**UPDATE** Dipendente **set** eMailUfficio = (**SELECT** ufficio.Email **FROM** PostazioneAttiva **JOIN** Ufficio **as** ufficio **on** PostazioneAttiva.Ufficio = ufficio.idUfficio **WHERE** PostazioneAttiva.idPostazioneAttiva = **NEW**.Postazione), isDaTrasferire = **FALSE** **WHERE** idDipendente = **NEW**.Dipendente;

**END**

**ImpiegoCorrente (Before Update):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`impiegoCorrente\_BEFORE\_UPDATE` **BEFORE** **UPDATE** **ON** `impiegoCorrente` **FOR** **EACH** **ROW**

**BEGIN**

**DECLARE** mansioneDipendente **int**;

**SET** mansioneDipendente = getMansioneUtente(**NEW**.Dipendente);

**IF** (**NEW**.Postazione != **OLD**.Postazione **AND** **OLD**.dummyValue = 0) **THEN**

**IF** **NOT** **EXISTS**(**SELECT** idPostazioneAttiva **FROM** PostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**JOIN** Mansione **on** Ufficio.Mansione = mansioneDipendente

**WHERE** (

**NOT** **EXISTS** (**SELECT** \* **FROM** ImpiegoPassato IP **WHERE** IP.dipendente = **NEW**.Dipendente **and** IP.postazione = **NEW**.Postazione)

**or**

**EXISTS** (**SELECT** \* **from** ImpiegoPassato IP **WHERE** IP.dipendente = **NEW**.Dipendente **and** IP.postazione = **NEW**.Postazione **and** IP.DataInizio < **DATE**(DATE\_sub(NOW(), **INTERVAL** 2 **YEAR**)))

)) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Il dipendente è stato già assegnato a questa postazione nei precedenti due anni!';

**END** **IF**;

**SET** **NEW**.DataInizio = NOW();

**INSERT** **INTO** ImpiegoPassato(DataInizio, DataFine, Postazione, Dipendente) **VALUES** (**OLD**.dataInizio, NOW(), **OLD**.Postazione, **OLD**.Dipendente);

**UPDATE** Dipendente **set** eMailUfficio = (**SELECT** ufficio.Email **FROM** PostazioneAttiva **JOIN** Ufficio **as** ufficio **on** PostazioneAttiva.Ufficio = ufficio.idUfficio **WHERE** PostazioneAttiva.idPostazioneAttiva = **NEW**.Postazione), isDaTrasferire = **FALSE** **WHERE** idDipendente = **NEW**.Dipendente;

**SET** **NEW**.dummyValue = 1;

**END** **IF**;

**IF** (**NEW**.Postazione != **OLD**.Postazione **AND** **OLD**.dummyValue != 0) **THEN**

**IF** **NOT** **EXISTS**(**SELECT** idPostazioneAttiva **FROM** PostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**JOIN** Mansione **on** Ufficio.Mansione = mansioneDipendente

**WHERE** (

**NOT** **EXISTS** (**SELECT** \* **FROM** ImpiegoPassato IP **WHERE** IP.dipendente = **NEW**.Dipendente **and** IP.postazione = **NEW**.Postazione)

**or**

**EXISTS** (**SELECT** \* **from** ImpiegoPassato IP **WHERE** IP.dipendente = **NEW**.Dipendente **and** IP.postazione = **NEW**.Postazione **and** IP.DataInizio < **DATE**(DATE\_sub(NOW(), **INTERVAL** 2 **YEAR**)))

**AND** Disponibile **is** **TRUE**) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Il dipendente è stato già assegnato a questa postazione nei precedenti due anni!';

**END** **IF**;

**SET** **NEW**.DataInizio = NOW();

**INSERT** **INTO** ImpiegoPassato(DataInizio, DataFine, Postazione, Dipendente) **VALUES** (**OLD**.dataInizio, NOW(), **OLD**.Postazione, **OLD**.Dipendente);

**UPDATE** PostazioneAttiva **set** Disponibile = **false** **WHERE** idPostazioneAttiva = **NEW**.Postazione;

**UPDATE** PostazioneAttiva **set** Disponibile = **true** **WHERE** idPostazioneAttiva = **OLD**.Postazione;

**UPDATE** Dipendente **set** eMailUfficio = (**SELECT** ufficio.Email **FROM** PostazioneAttiva **JOIN** Ufficio **as** ufficio **on** PostazioneAttiva.Ufficio = ufficio.idUfficio **WHERE** PostazioneAttiva.idPostazioneAttiva = **NEW**.Postazione), isDaTrasferire = **FALSE** **WHERE** idDipendente = **NEW**.Dipendente;

**END** **IF**;

**IF** (**NEW**.Dipendente != **OLD**.Dipendente) **THEN**

**IF** **NOT** **EXISTS** (**SELECT** 1 **FROM** Dipendente **WHERE** idDipendente = **NEW**.Dipendente **and** isDaTrasferire **is** **true**) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'L\'ID del dipendente non è valido o non disponibile per il trasferimento!';

**END** **IF**;

**END** **IF**;

**END**

**PostazioneAttiva (Before Insert):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`PostazioneAttiva\_BEFORE\_INSERT` **BEFORE** **INSERT** **ON** `PostazioneAttiva` **FOR** **EACH** **ROW**

**BEGIN**

**IF** **NOT** **EXISTS** (**SELECT** 1 **FROM** Postazione **WHERE** idPostazione = **NEW**.idPostazioneAttiva **and** Ufficio != 0) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Nessuna postazione associata all\'ID indicato!';

**END** **IF**;

**END**

**PostazioneAttiva (After Insert):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`PostazioneAttiva\_AFTER\_INSERT` **AFTER** **INSERT** **ON** `PostazioneAttiva` **FOR** **EACH** **ROW**

**BEGIN**

**IF** **NEW**.Disponibile **is** **true** **THEN**

**UPDATE** Ufficio **set** NumeroPostazioniLibere = NumeroPostazioniLibere + 1 **WHERE** idUfficio = **NEW**.Ufficio;

**ELSE**

**UPDATE** Ufficio **set** NumeroPostazioniLibere = NumeroPostazioniLibere - 1 **WHERE** idUfficio = **NEW**.Ufficio;

**END** **IF**;

**END**

**PostazioneAttiva (After Update):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`PostazioneAttiva\_AFTER\_UPDATE\_1` **AFTER** **UPDATE** **ON** `PostazioneAttiva` **FOR** **EACH** **ROW**

**BEGIN**

**IF** (**OLD**.Disponibile != **NEW**.Disponibile **and** **NEW**.Disponibile = **false**) **then**

**UPDATE** Ufficio **set** NumeroPostazioniLibere = NumeroPostazioniLibere - 1 **WHERE** idUfficio = **NEW**.Ufficio;

**END** **IF**;

**IF** (**OLD**.Disponibile != **NEW**.Disponibile **and** **NEW**.Disponibile = **true**) **then**

**UPDATE** Ufficio **set** NumeroPostazioniLibere = numeroPostazioniLibere + 1 **Where** idUfficio = **NEW**.Ufficio;

**END** **IF;**

**END**

**PostazioneAttiva (Before Delete):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`PostazioneAttiva\_BEFORE\_DELETE` **BEFORE** **Delete** **ON** `PostazioneAttiva` **FOR** **EACH** **ROW**

**BEGIN**

**IF** (**OLD**.Disponibile **is** **false**) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Non è possibile disabilitare una postazione attualmente assegnata ad un dipendente!';

**END** **IF**;

**UPDATE** Ufficio **SET** NumeroPostazioniLibere = NumeroPostazioniLibere - 1 **WHERE** idUfficio = **OLD**.Ufficio;

**END**

**Postazione (Before Update):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`Postazione\_BEFORE\_UPDATE` **BEFORE** **UPDATE** **ON** `Postazione` **FOR** **EACH** **ROW**

**BEGIN**

**IF** (**NEW**.Ufficio != **OLD**.Ufficio) **THEN**

**IF** **EXISTS** (**SELECT** 1 **FROM** PostazioneAttiva **WHERE** idPostazioneAttiva = **OLD**.idPostazione **and** Disponibile **is** **false**) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Non è assegnare spostare una postazione attualmente occupata!';

**END** **IF**;

**IF** **NOT** **EXISTS** (**SELECT** 1 **FROM** Ufficio **WHERE** idUfficio = **NEW**.Ufficio) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'L\'ID dell\'ufficio indicato non è corretto!';

**END** **IF**;

**END** **IF**;

**END**

**Postazione (Before Delete):**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`Postazione\_BEFORE\_DELETE` **BEFORE** **DELETE** **ON** `Postazione` **FOR** **EACH** **ROW**

**BEGIN**

**IF** **EXISTS** (**SELECT** 1 **FROM** PostazioneAttiva **WHERE** idPostazioneAttiva = idPostazione **and** Disponibile **is** **false**) **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Non è possibile eliminare una postazione attualmente occupata!';

**END** **IF**;

**END**

**CREATE** **DEFINER**=`root`@`localhost` **TRIGGER** `mydb`.`Dipendente\_BEFORE\_UPDATE` **BEFORE** **UPDATE** **ON** `Dipendente` **FOR** **EACH** **ROW**

**BEGIN**

**IF** (**NEW**.Mansione != **OLD**.Mansione) **THEN**

**SET** **NEW**.isDaTrasferire = **TRUE**;

**END** **IF**;

**END**

## Viste

**GetPostazioni:** la vista permette di ottenere la lista delle postazioni attualmente presenti nella base di dati.

**CREATE** **OR** **REPLACE**

**VIEW** `getPostazioni` **AS** **select**

`Postazione`.`idPostazione` **AS** `idPostazione`,

`Postazione`.`Ufficio` **AS** `Ufficio`

**from**

`Postazione`

**GetPostazioniAttive:** la vista permette di ottenere la lista delle postazioni attualmente attive presenti nelal base di dati.

**CREATE** **OR** **REPLACE**

**VIEW** `getPostazioniAttive` **AS** **select**

`PostazioneAttiva`.`idPostazioneAttiva` **AS** `idPostazioneAttiva`,

`PostazioneAttiva`.`Disponibile` **AS** `Disponibile`,

`PostazioneAttiva`.`Ufficio` **AS** `Ufficio`

**from**

`PostazioneAttiva`

**getToTransfer:** la vista permette di ottenere la lista dei dipendenti, presenti nella base di dati, che devono essere trasferiti.

**CREATE** **OR** **REPLACE**

**VIEW** `getToTransfer` **AS** **select**

`Dipendente`.`idDipendente` **AS** `idDipendente`

**from**

`Dipendente`

**where**

`mydb`.`Dipendente`.`daTrasferire` **is** **true**

## Stored Procedures e transazioni

Mostrare e commentare le stored procedure che sono state realizzate per implementare la logica applicativa delle operazioni sui dati, evidenziando quando (e perché) sono state realizzate operazioni transazionali complesse.

**MakeScambioUser:** sono state utilizzate operazioni transazionali complesse per assicurare la consistenza delle informazioni all’interno del database in modo, come in questo caso, di non “lasciare senza nessuna mansione I dipendenti”

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`makeScambioUser`(user1 **int**, user2 **int**)

**BEGIN**

**DECLARE** postazione1 **int**;

**DECLARE** postazione2 **int**;

**DECLARE** validUser1 **int**;

**DECLARE** validUser2 **int**;

**DECLARE** **EXIT** HANDLER **FOR** **SQLEXCEPTION**

**BEGIN**

**ROLLBACK**;

**END**;

**SET** validUser1 = checkValidUser(user1);

**SET** validUser2 = checkValidUser(user2);

**SET** postazione1 = getCurrentPostazione(user1);

**SET** postazione2 = getCurrentPostazione(user2);

**IF** (postazione1 **is** **not** **null** **and** postazione2 **is** **not** **null**) **THEN**

**UPDATE** impiegoCorrente **SET** impiegoCorrente.dummyValue = 0 **WHERE** impiegoCorrente.Dipendente = user1;

**UPDATE** impiegoCorrente **SET** impiegoCorrente.Postazione = postazione2 **WHERE** impiegoCorrente.Dipendente = user1;

**UPDATE** impiegoCorrente **SET** impiegoCorrente.dummyValue = 0 **WHERE** impiegoCorrente.Dipendente = user2;

**UPDATE** impiegoCorrente **SET** impiegoCorrente.Postazione = postazione1 **WHERE** impiegoCorrente.Dipendente = user2;

**ELSE**

**ROLLBACK**;

**END** **IF**;

**COMMIT**;

**END**

**AssignPostazioneToUfficio:** viene utilizzata un’operazione transazionale complessa in quanto è necessario l’aggiornamento dell’informazione relativa alla stessa postazione ma di due tabelle differenti

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`assignPostazioneToUfficio`(**IN** postazione **Int**, **IN** ufficioID **Int**)

**BEGIN**

**DECLARE** validPostazione **INT**;

**DECLARE** **EXIT** HANDLER **FOR** **SQLEXCEPTION**

**BEGIN**

**ROLLBACK**;

**END**;

**IF** **EXISTS** (**SELECT** 1 **FROM** Postazione **WHERE** Postazione.ID = postazione **and** Postazione.Ufficio = ufficioID) **then**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Si sta assegnando al dipendente la sua mansione attuale!';

**END** **IF**;

**SET** validPostazione = checkValidPostazione(postazione);

**SET** FOREIGN\_KEY\_CHECKS = 0;

**UPDATE** PostazioneAttiva **SET** PostazioneAttiva.Ufficio = ufficioID **WHERE** PostazioneAttiva.idPostazione = postazione;

**UPDATE** Postazione **SET** Postazione.Ufficio = ufficioID **WHERE** Postazione.idPostazione = postazione;

**SET** FOREIGN\_KEY\_CHECKS = 1;

**COMMIT**;

**END**

**AssignToPostazioneFree:** viene utilizzata un’operazione transazionale complessa in quanto è necessario l’aggiornamento dell’informazione relativa alla stessa postazione ma di due tabelle differenti

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`assignToPostazioneFree`(**IN** dipendente **Int**, **IN** newPostazione **Int**)

**BEGIN**

**DECLARE** mansioneDipendente **Int**;

**DECLARE** littleResult **INT**;

**DECLARE** postazioneDipendente **INT**;

**DECLARE** validUser **INT**;

**DECLARE** **EXIT** HANDLER **FOR** **SQLEXCEPTION**

**BEGIN**

**ROLLBACK**;

**END**;

**SET** postazioneDipendente = getCurrentPostazione(dipendente);

**SET** mansioneDipendente = getMansioneUtente(dipendente);

**SET** littleResult = checkPostazione(newPostazione);

**SET** validUser = checkValidUser(dipendente);

**IF** postazioneDipendente **is** **null** **then**

**INSERT** **INTO** impiegoCorrente(DataInizio, Postazione, Dipendente) **VALUES** (NOW(), newPostazione, dipendente);

**UPDATE** Dipendente **SET** isDaTrasferire = **FALSE** **WHERE** idDipendente = dipendente;

**END** **IF**;

**IF** postazioneDipendente **is** **not** **null** **THEN**

**UPDATE** impiegoCorrente **SET** impiegoCorrente.Postazione = newPostazione **WHERE** impiegoCorrente.Dipendente = dipendente;

**UPDATE** Dipendente **SET** isDaTrasferire = **FALSE** **WHERE** idDipendente = dipendente;

**END** **IF**;

**COMMIT**;

**END**

**CheckPostazioniDisponibili:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`checkPostazioniDisponibili`(**IN** newDipendente **Int**)

**BEGIN**

**DECLARE** postazioneDipendente **Int**;

**DECLARE** mansioneDipendente **Int**;

**DECLARE** counter **int**;

**DECLARE** postazioneExists **int**;

/\* Controllo che l'utenta esista e che sia indicato come da trasferire \*/

**SET** counter = checkValidUser(newDipendente);

**SET** mansioneDipendente = getMansioneUtente(newDipendente);

**SET** postazioneDipendente = getCurrentPostazione(newDipendente);

**SELECT** Postazione **INTO** postazioneDipendente **from** impiegoCorrente **where** Dipendente = newDipendente;

**IF** postazioneDipendente **is** **null** **then**

**SELECT** idPostazioneAttiva **FROM** PostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**JOIN** Mansione **on** Ufficio.Mansione = mansioneDipendente

**WHERE** (

**NOT** **EXISTS** (**SELECT** \* **FROM** ImpiegoPassato IP **WHERE** IP.dipendente = newDipendente **and** IP.postazione = idPostazioneAttva)

**or**

**EXISTS** (**SELECT** \* **from** ImpiegoPassato IP **WHERE** IP.dipendente = newDipendente **and** IP.postazione = idPostazioneAttiva **and** IP.DataInizio < **DATE**(DATE\_sub(NOW(), **INTERVAL** 2 **YEAR**)))

)

**AND** Disponibile **is** **TRUE**;

**ELSE**

**SELECT** idPostazioneAttiva **FROM** PostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**JOIN** Mansione **on** Ufficio.Mansione = mansioneDipendente

**WHERE** idPostazioneAttiva != postazioneDipendente

**AND** (

**NOT** **EXISTS** (**SELECT** \* **FROM** ImpiegoPassato IP **WHERE** IP.dipendente = newDipendente **and** IP.postazione = idPostazioneAttiva)

**or**

**EXISTS** (**SELECT** \* **from** ImpiegoPassato IP **WHERE** IP.dipendente = newDipendente **and** IP.postazione = idPostazioneAttiva **and** IP.DataInizio < **DATE**(DATE\_sub(NOW(), **INTERVAL** 2 **YEAR**)))

)

**AND** Disponibile **is** **TRUE**;

**END** **IF**;

**END**

**DeletePostazione:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`deletePostazione`(**IN** postazione **Int**)

**BEGIN**

**DECLARE** validPostazione **INT**;

**SET** validPostazione = checkValidPostazione(postazione);

**DELETE** **FROM** Postazione **WHERE** idPostazione = postazione;

**END**

**DisablePostazione:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`disablePostazione`(**IN** postazione **Int**)

**BEGIN**

**DECLARE** validPostazione **INT**;

/\* Aggiungere per il controllo se la posizione poteva essere disabilitata \*/

/\* Anche qui aggiungere il check \*/

**SET** validPostazione = checkValidPostazione(postazione);

**DELETE** **FROM** PostazioneAttiva **WHERE** idPostazioneAttiva = postazione;

**END**

**EditMansione:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`editMansione`(**IN** userID **int**, **IN** idMansione **int**)

**BEGIN**

**DECLARE** validUser **int**;

**DECLARE** test **int**;

**set** validUser = checkUserExists(userID);

/\* Al momento è implementato nel trigger, ed è Ok.

\* Il controllo del dipendente però va fatto prima, lui comunque non mi da errore

\* Però, se ho il check di una foreign key, lo potrei evitare \*/

**SET** test = checkValidMansione(idMansione);

**IF** **EXISTS** (**SELECT** 1 **FROM** Dipendente **WHERE** idDipendente = userID **and** Mansione = idMansione) **then**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Si sta assegnando al dipendente la sua mansione attuale!';

**END** **IF**;

**UPDATE** Dipendente **SET** Mansione = idMansione **WHERE** idDipendente = userID;

**END**

**EnablePostazione:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`enablePostazione`(**IN** postazione **Int**)

**BEGIN**

**DECLARE** validPostazione **INT**;

**DECLARE** idUfficio **int**;

**SET** validPostazione = checkValidPostazione(postazione);

**IF** **EXISTS** (**SELECT** 1 **FROM** PostazioneAttiva **WHERE** idPostazioneAttiva = postazione) **then**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'La postazione risulta già abilitata!';

**END** **IF**;

**SELECT** Ufficio **into** idUfficio **from** Postazione **where** idPostazione = postazione;

**INSERT** **INTO** PostazioneAttiva(Disponibile, idPostazioneAttiva, Ufficio, idPostazione)

**VALUES** (**true**, postazione, idUfficio, postazione);

**END**

**GetAssegnazioniPassate**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`getAssegnazioniPassate`(**IN** idUtente **int**)

**BEGIN**

/\* Controllo l'esistenza del dipendente \*/

**DECLARE** validUtente **int**;

**SET** validUtente = checkUserExists(idUtente);

**SELECT** idImpiegoPassato, DataInizio, DataFine, Postazione **FROM** ImpiegoPassato **WHERE** Dipendente = idUtente;

**END**

**GetPostazioni**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`getPostazioni`()

**BEGIN**

**SELECT** \* **from** getPostazioni;

**END**

**GetPostazioniAttive**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`getPostazioniAttive`()

**BEGIN**

**SELECT** \* **from** getPostazioniAttive;

**END**

**GetUserDaTrasferire:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`getUserDaTrasferire`()

**BEGIN**

**SELECT** \* **from** getToTransfer;

**END**

**GetUserNumeroInterno:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`getUserNumeroInterno`(**IN** numeroInterno **int**)

**BEGIN**

/\* Devo prima mostrare le informazioni relative all'ufficio \*/

**SELECT** Ufficio.idUfficio, Ufficio.Piano, Ufficio.Edificio, Ufficio.Mansione

**FROM** Ufficio

**WHERE** Ufficio.TelefonoInterno = numeroInterno;

/\* Successivamente devo mostrare le informazioni relative ai dipendenti che stanno dentro

\* l'ufficio

\* \*/

**SELECT** Dipendente.idDipendente, Dipendente.isDaTrasferire

**FROM** impiegoCorrente

**JOIN** PostazioneAttiva **on** impiegoCorrente.Postazione = PostazioneAttiva.idPostazioneAttiva

**JOIN** Ufficio **on** Ufficio.idUfficio = PostazioneAttiva.Ufficio

**JOIN** Dipendente **on** Dipendente.idDipendente = impiegoCorrente.Dipendente

**WHERE** Ufficio.TelefonoInterno = numeroInterno;

**END**

**GetUserScambiabili:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`getUserScambiabili`(`newUser` **int**)

**BEGIN**

**DECLARE** `newMansione` **int**;

**DECLARE** `newPostazione` **int**;

/\* Effettuo il controllo sull'esistenza ed il trasferire dell'utente \*/

**DECLARE** `validUser` **int**;

**SET** validUser = checkValidUser(newUser);

**SET** newPostazione = getCurrentPostazione(newUser);

**SET** newMansione = getMansioneUtente(newUser);

**SELECT** I2.dipendente **as** idImpiegato, I2.postazione **as** idPostazione

**from** impiegoCorrente I2

**JOIN** Dipendente **on** Dipendente.idDipendente = I2.Dipendente

**WHERE**

(

(

**NOT** **EXISTS** (**SELECT** \* **FROM** ImpiegoPassato IP **WHERE** IP.dipendente = newUser **and** IP.postazione = I2.postazione)

**or**

**EXISTS** (**SELECT** \* **from** ImpiegoPassato IP **WHERE** IP.dipendente = newUser **and** IP.postazione = I2.postazione **and** IP.DataInizio < **DATE**(DATE\_sub(NOW(), **INTERVAL** 2 **YEAR**)))

)

**and**

(

**NOT** **EXISTS**(**SELECT** \* **FROM** ImpiegoPassato IP **WHERE** IP.Dipendente = I2.Dipendente **and** IP.postazione = newPostazione)

**or**

**EXISTS**(**SELECT** \* **from** ImpiegoPassato IP **WHERE** IP.Dipendente = I2.Dipendente **and** IP.postazione = newPostazione **and** IP.DataInizio < **DATE**(DATE\_sub(NOW(), **INTERVAL** 2 **YEAR**)))

)

**and** I2.dipendente != newUser **and** Dipendente.Mansione = newMansione

**and** Dipendente.isDaTrasferire = **true**

**and** **EXISTS** (**SELECT** Dipendente **FROM** impiegoCorrente **WHERE** Dipendente = I2.Dipendente)

);

**END**

**InsertNewPostazione:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`insertNewPostazione`(**IN** ufficio **INT**, **OUT** idPostazione **INT**)

**BEGIN**

**DECLARE** validPostazione **INT**;

**INSERT** **INTO** Postazione(Ufficio) **VALUES** (ufficio);

**SET** idPostazione = **LAST\_INSERT\_ID**();

**END**

**LoginUser:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`loginUser`(**IN** matricolaDipendente **int**, **IN** passwordDipendente **Varchar**(256), **OUT** tipoDipendente **INT**)

**BEGIN**

**SELECT** userType **INTO** tipoDipendente **FROM** Dipendente **WHERE** idDipendente = matricolaDipendente **AND** password = SHA2(passwordDipendente, 256);

**IF** tipoDipendente **is** **NULL** **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Nome utente e/o password errati!';

**END** **IF**;

**END**

**SetUserDaTrasferire:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`setUserDaTrasferire`(**IN** utente **INT**)

**BEGIN**

**UPDATE** Dipendente **SET** isDaTrasferire = **TRUE** **Where** idDipendente = utente;

**END**

**UserNameSurname:**

**CREATE** **DEFINER**=`root`@`localhost` **PROCEDURE** `mydb`.`userNameSurname`(name **VARCHAR**(255), surname **VARCHAR**(255))

**BEGIN**

**IF** (name = '' **and** surname = '') **THEN**

signal **sqlstate** '45000'

**SET** MESSAGE\_TEXT = 'Inserire nome e/o cognome del dipendente!';

**ELSE**

**IF** (surname = '') **THEN**

**SELECT** Dipendente.EMailUfficio, Dipendente.EMailPersonale, Ufficio.idUfficio, Ufficio.Piano, Ufficio.Edificio **FROM** Dipendente

**JOIN** impiegoCorrente **ON** impiegoCorrente.Dipendente = Dipendente.idDipendente

**JOIN** PostazioneAttiva **on** impiegoCorrente.Postazione = PostazioneAttiva.idPostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**WHERE** Dipendente.Nome = name;

**END** **IF**;

**IF** (name = '') **THEN**

**SELECT** Dipendente.EMailUfficio, Dipendente.EMailPersonale, Ufficio.idUfficio, Ufficio.Piano, Ufficio.Edificio **FROM** Dipendente

**JOIN** impiegoCorrente **ON** impiegoCorrente.Dipendente = Dipendente.idDipendente

**JOIN** PostazioneAttiva **on** impiegoCorrente.Postazione = PostazioneAttiva.idPostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**WHERE** Dipendente.Cognome = surname;

**END** **IF**;

**IF** (name != '' **and** surname != '') **THEN**

**SELECT** Dipendente.EMailUfficio, Dipendente.EMailPersonale, Ufficio.idUfficio, Ufficio.Piano, Ufficio.Edificio **FROM** Dipendente

**JOIN** impiegoCorrente **ON** impiegoCorrente.Dipendente = Dipendente.idDipendente

**JOIN** PostazioneAttiva **on** impiegoCorrente.Postazione = PostazioneAttiva.idPostazioneAttiva

**JOIN** Ufficio **on** PostazioneAttiva.Ufficio = Ufficio.idUfficio

**WHERE** Dipendente.Cognome = surname **AND** Dipendente.Nome = name;

**END** **IF**;

**END** **IF**;

**END**

# Appendice: Implementazione

## Codice SQL per instanziare il database

Riportare il codice SQL necessario ad istanziare lo schema del DB.

## Codice del Front-End

Riportare (correttamente formattato) il codice C del thin client realizzato per interagire con la base di dati.

1. Indicare con E le entità, con R le relazioni [↑](#footnote-ref-2)
2. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-3)
3. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-4)
4. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-5)
5. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-6)
6. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-7)
7. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-8)
8. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-9)