Gestione locali e risorse università

di Zecchini Samuele e Verucchi Micaela
CDL INFORMATICA AA2013/2014

Indice:

1-Interpretazione traccia;

2-Glossario base di dati;

3-Vincoli aggiuntivi;

4-Progetto concettuale passo-passo;

5-Progetto concettuale completo;

6-Progetto logico-schema relazionale;

7-SQL creazione tabelle e popolamento base di dati;

8-Operazioni ;

9-SQL operazioni;

10-Dati derivati;

11-Trigger;

1- Interpretazione del testo

Si vuole realizzare un'applicazione database per gestire le informazioni relative all'utilizzo dei locali e delle risorse di un'Università da parte degli studenti dei Corsi di Laurea e del personale, docente e non docente, dell'Università stessa.
Gli accessi ad alcuni locali dell'Università sono riservati a persone autorizzate in base alle seguenti modalità. Un utente può avere per un locale un solo permesso. Ogni permesso appartiene ad una certa tipologia che specifica tra l'altro, per ciascuno dei sei giorni lavorativi (ipotizziamo che l'intervallo di accesso sia lo stesso per ogni giorno della settimana), gli intervalli temporali di accesso. Gli accessi sono controllati e rilevati tramite la lettura di una tessera magnetica assegnata a ciascun utente.
Vi sono alcuni locali per cui bisogna tenere traccia di tutti gli accessi, rilevando per ogni utente: data, ora e identificazione dell'utente. Per altri locali bisogna tenere traccia solo del numero di accessi giornalieri. Tutte le volte che il sistema rifiuta l'accesso ad un locale, si deve tenere traccia dell'evento, memorizzando tutte le informazioni relative. In particolare, interessa evidenziare i casi in cui per uno stesso utente e per lo stesso locale ci siano stati più di tre rifiuti giornalieri.
Tra i locali dell'Università vi sono dei laboratori didattici che contengono un insieme di posti di lavoro (un posto di lavoro è visto come una singola postazione) ed un insieme di risorse. Ad ogni posto di lavoro sono assegnate alcune risorse (quali ad esempio, unità di calcolo, stampanti, software applicativo). Alcuni posti di lavoro sono resi disponibili a tutti gli studenti senza controlli, altri vengono mensilmente assegnati a particolari studenti, quali ad esempio laureandi (sono stati considerati solo laureandi), previa autorizzazione di un docente. Infine, in un laboratorio vi possono essere un certo numero di posti di lavoro prenotabili giornalmente da parte di un singolo studente oppure da parte di un docente titolare di un insegnamento; per questi posti di lavoro si deve tenere traccia di tutte le prenotazioni e di tutti le utilizzazioni da parte degli studenti. Per le prenotazioni devono essere garantite le seguenti regole di non sovrapposizione:

1. in una certa ora, un posto può essere prenotato da un solo studente;
2. in una certa ora, uno studente non può avere la prenotazione per due posti differenti.

Ogni laboratorio ha come responsabile organizzativo un docente e come responsabile operativo un tecnico dell'Università.
Alcune unità di calcolo richiedono un account per accedervi; gli account vengono rilasciati automaticamente, su un insieme di server, agli studenti sulla base della loro iscrizione ad un corso d+i laurea e all'anno di iscrizione. A ciascuno dei gruppi così individuato il System Administrator (non vi è un solo System Administrator ma tale operazione può essere eseguita da più tecnici che siano System Administrator) assegna e gestisce determinati privilegi e risorse. Per ogni account viene gestita la configurazione del profilo utente, memorizzando informazioni quali lo "shell" e gli applicativi utilizzati (considerate risorse assegnate al gruppo di account e non al singolo).

2 – Glossario base di dati

TERMINE	DESCRIZIONE	SINONIMI	LEGAMI
Utente	Nome Cognome Data di Nascita Luogo di Nascita Residenza CAP Nazionalità Codice Fiscale		Studente Personale Tessera Magnetica Permesso Prenotazione
Studente	Matricola		Account Utente Laureando Liberi
Personale	Salario		Utente Docente Non Docente
Laureando			Studente Assegnazione
Docente	Titolare	Docente titolare Docente non titolare	Personale Laboratorio Assegnazione Prenotazione
Non docente			Personale Tecnico
Tecnico	Ruolo	System Administrator	Non docente Laboratorio DistribuzionePR
Corso di laurea	Id corso Nome		Gruppo di Account
Locale	Id Locale Nome Luogo Dimensioni Capacità		Permesso Laboratorio Accesso Rifiuto
Laboratorio	Sede		Locale Posto di Lavoro Docente Tecnico

Permesso	Id Permesso		Utente Tipologia Locale
Tipologia permesso	Id Tipologia Intervallo accesso Giorno		Permesso
Accesso	Id Accesso Data Ora		Tessera Magnetica Utente Locale
Rifiuto	Id Rifiuto Data Ora		Tessera magnetica Locale Utente
Tessera Magnetica	Id Tessera		Utente Accesso Rifiuto
Posto di lavoro	Id Posto		Risorsa Laboratorio Libero Assegnato Prenotabile
Prenotabile			Utente Posto di Lavoro Prenotazione
Assegnabile			Laureando Docente Posto di Lavoro Assegnazione
Risorsa	Id Risorsa Nome	“Shell” Software applicativi Privilegi Stampante Computer DPI Beker Microscopio Vetrini Ecc…	Posto di Lavoro DistribuzionePR

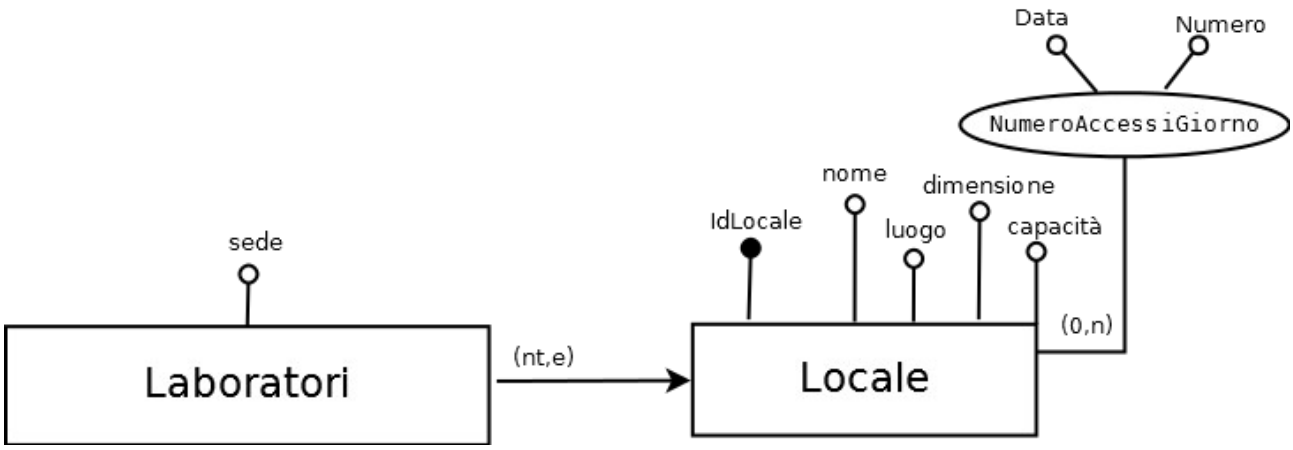
Prenotazione	Data Ora	Prenotabile Utente
Assegnazione	Mese	Docente Assegnabile Laureando
Account	User Password	Studente Gruppo di Account
Gruppo di Account	Anno Corso	DistribuzionePR Account Corso di Laurea
DistribuzionePR		Gruppo di Account Risorsa Tecnico

3 -Vincoli aggiuntivi:

1. Un tecnico può gestire un solo laboratorio;
2. Un laboratorio deve essere gestito da solo un tecnico;
3. Un laboratorio può avere massimo 100 posti di lavoro;
4. Un laureando al mese può prenotare un solo laboratorio;
5. Un posto di lavoro può avere massimo 20 risorse;
6. Le risorse utilizzate sono assegnate al gruppo account invece che al singolo account, da parte di uno o più tecnici;

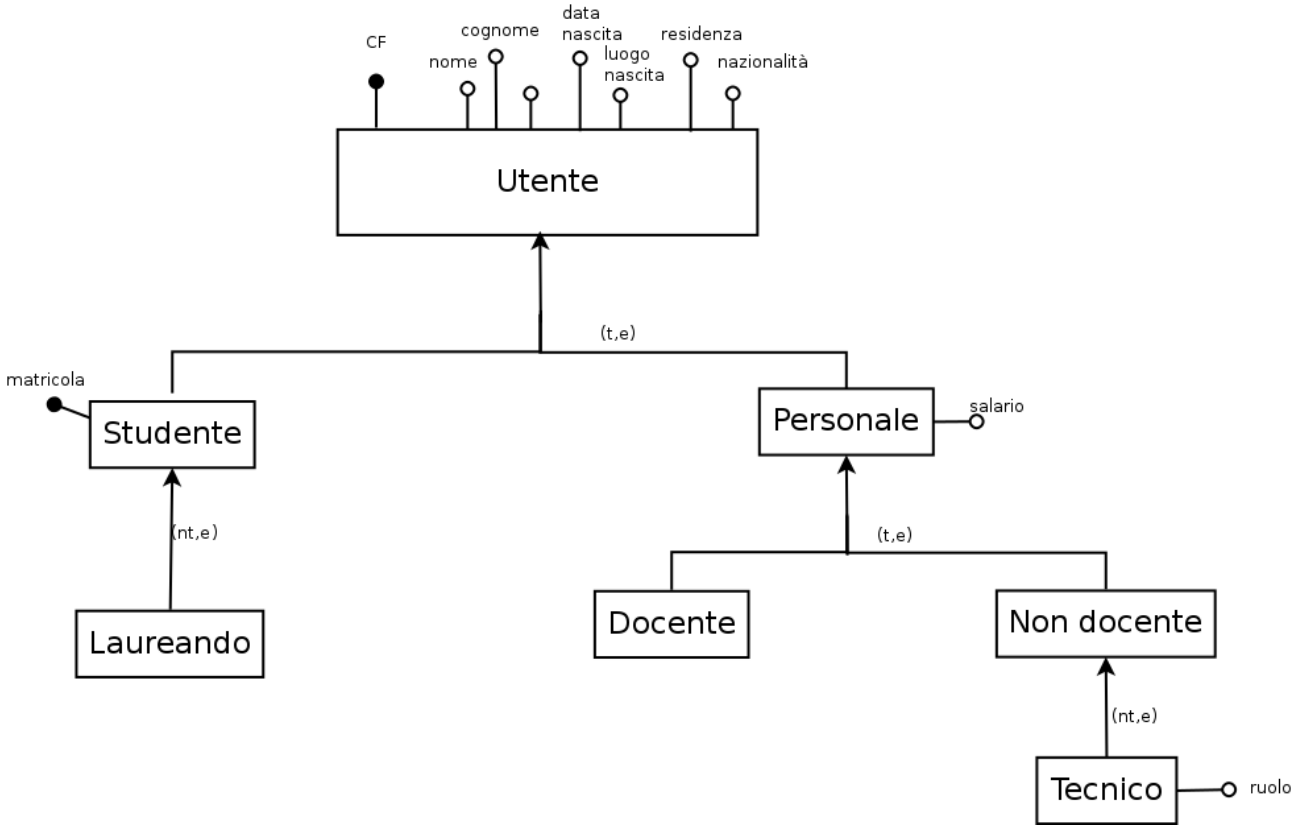
4-Progetto concettuale passo-passo;

Gerarchia Locale:



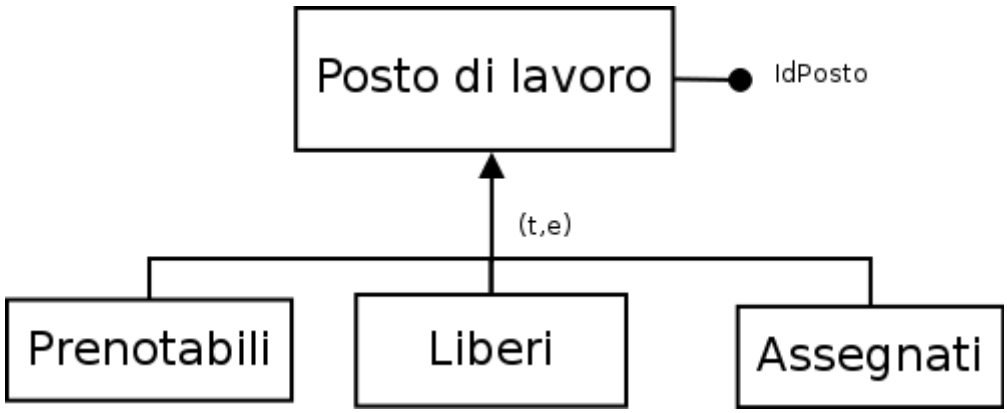
In questa gerarchia troviamo la definizione dell'entità “Locale”, e della sua entità figlia “Laboratori”. Il locale ha vari attributi che lo descrivono, tra cui l'idLocale che lo identifica in maniera univoca e un attributo composto multiplo NumeriAccessi che conteggia il numero di accessi giornalieri.
Infatti abbiamo pensato di non suddividere i locali in più tipologie (eccetto quella necessaria dei laboratori), per cui non vi è la differenza esplicita di locali con permesso o meno, o locali che hanno solo un conteggio giornaliero o tutti i dati degli accessi. Tutti potenzialmente potrebbero registrare tali dati.
La classe figlia Laboratori è una classe che identifica tutti i possibili laboratori presenti in un'università: quello di informatica, quello di chimica, biologia ecc. Nel laboratorio il campo capacità non indica il numero massimo di posti di lavoro ma il numero massimo di persone ammesse all'interno di esso.
La gerarchia è non totale in quanto i laboratori sono solo un sottoinsieme dei Locali, ci possono essere Locali che non sono Laboratori, ma che è interessante memorizzare nella base di dati; essa è inoltre esclusiva in quanto Laboratori è l'unica classe figlia in questa gerarchia.

Gerarchia Utente:



Tale gerarchia descrive l'utenza dei locali universitari, accorpati complessivamente in Utente e specificati nelle classi figlie. L'Utente è identificato dal codice fiscale e possiede altri tipici dati anagrafici.
Scendendo al primo livello della gerarchia troviamo le entità Studente e Personale. Questo primo livello della gerarchia è totale ed esclusivo: totale in quanto si presuppone che un utente sia per forza o uno studente oppure un elemento del personale universitario; esclusiva perchè non ci possono essere studenti che appartengono anche al personale.
La classe Studente contiene un attributo chiave matricola che lo identifica univocamente; tale entità è inoltre maggiormente specificata dall'entità figlia Laureando, la quale descrive un particolare tipo di studente che ha privilegi diversi dallo studente generico. La gerarchia è non totale in quanto ci sono studenti non laureandi ed è esclusiva.
L'entità Personale, che ha come attributo aggiuntivo un salario, viene ulteriormente specificata dalle entità “Docente” e “Non docente”. Docente contiene tutti gli insegnanti universitari, sia titolari di corsi che meno (differenziati dal flag “titolare”), mentre Non docente contiene tutto il personale che non insegna ed è ulteriormente specificato da Tecnico. Tra i tecnici vi è anche il System Administrator, identificabile dall'attributo ruolo.
La gerarchia Personale, Docente e Non docente è totale ed esclusiva, mentre quella tra Non docente e Tecnico è non totale ma esclusiva.

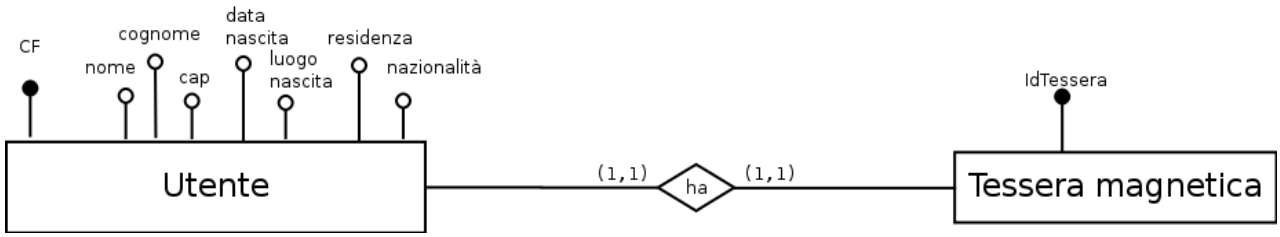
Gerarchia Posti di lavoro:



Questa gerarchia gestisce i posti di lavoro.
Il concetto di posto di lavoro è stato interpretato come una postazione all'interno di un laboratorio dove un certo utente, studente o docente che sia, possa utilizzare una qualsivoglia risorsa(vedi associazione posto di lavoro->risorsa e Posto di lavoro->laboratorio). Questi "Posti" però si distinguono in varie categorie tutti aventi una rilevante importanza all'interno del problema.
L'entità padre viene specificata da tre entità figlie attraverso una gerarchia totale ed esclusiva, tra le entità figlie troviamo:

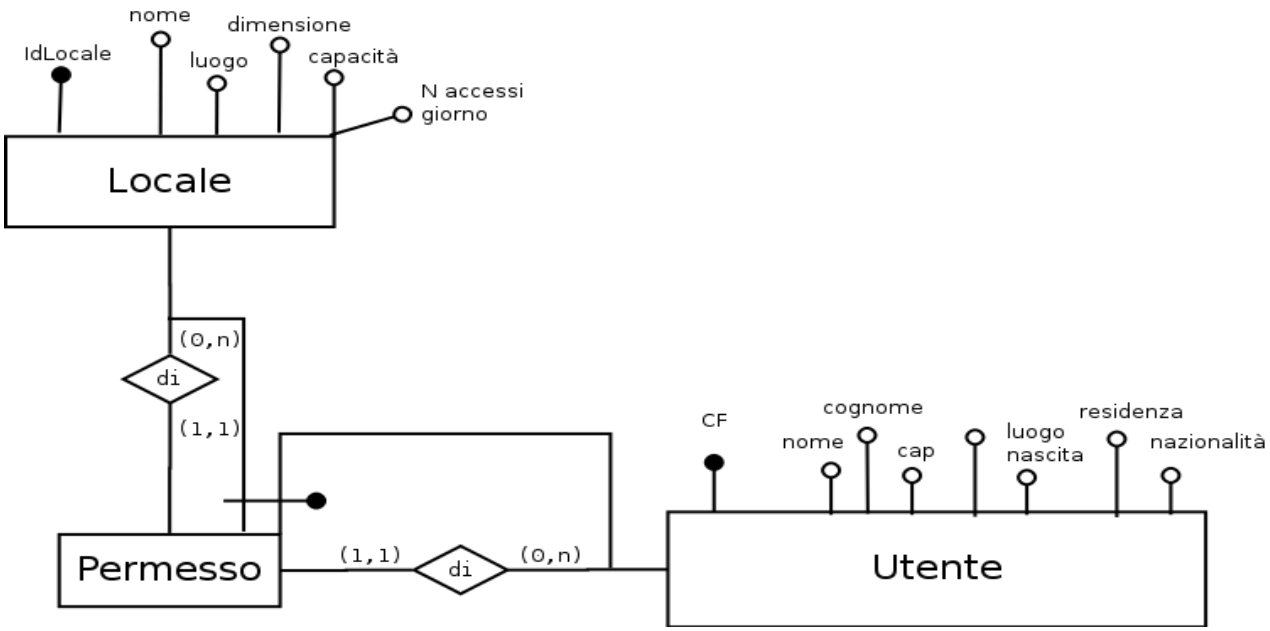
- Prenotabili: questa entità descrive un particolare tipo di posto di lavoro che è prenotabile giornalmente da un docente titolare di un corso o da uno studente.
- Liberi: descrive i posti liberamente usufruibili da uno studente senza prenotazione.
- Assegnati: particolare tipo di posto di lavoro assegnabile mensilmente solo ai Laureandi previa autorizzazione di un docente(vedi associazione assegnati->laureandi e assegnati->docente).

Associazione Tessera Magnetica-Utente:



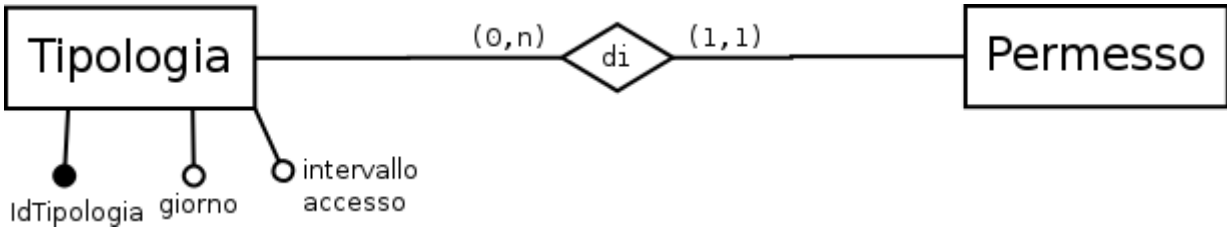
Questa associazione mette in evidenza la relazione che c'è tra un Utente e una Tessera magnetica. L'entità Tessera magnetica descrive il mezzo attraverso il quale un utente può accedere ad un locale. Essa ha un Id che la identifica in maniera univoca ed è associata ad Utente con cardinalità (1,1) in quanto una tessera può essere di un solo Utente ma allo stesso modo anche la cardinalità Utente-Tessera è (1,1) perchè un utente deve possedere una ed una sola tessera.

Entità permesso:



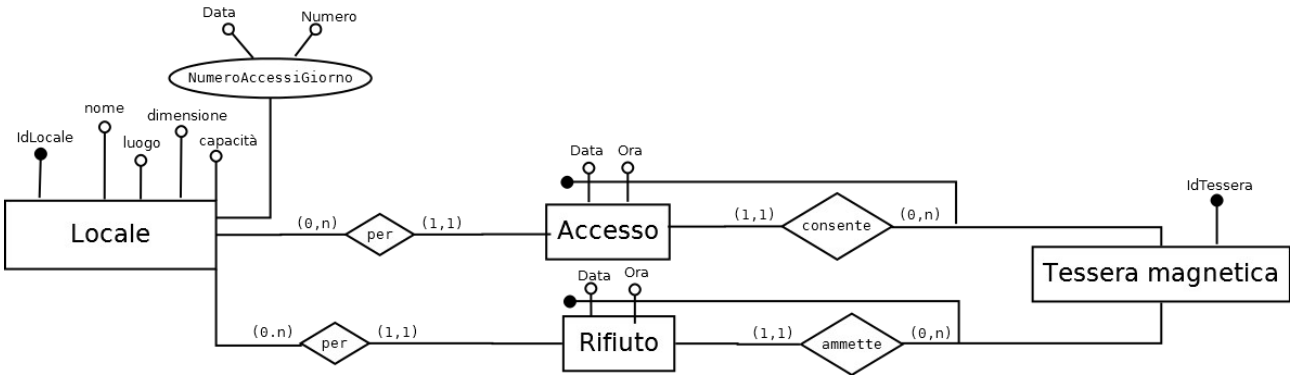
Un utente può accedere ad un certo locale se e soltanto se ha il “permesso di farlo”.
Questo concetto di Permesso rappresenta l'autorizzazione che un certo utente ha per un certo locale. Tale permesso può essere di varie tipologie(vedi associazione successiva) divise in base alla loro fascia oraria di accesso.
Si può notare come la chiave di Permesso si la composizione tra la chiave di utente e la chiave di locale, tramite un doppia identificazione esterna, questa soluzione permette di stabilire che un utente per un certo locale può avere un solo permesso, come richiesto dal problema.

Associazione Permesso-Tipologia:



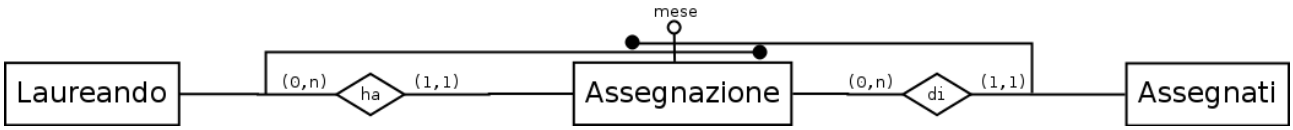
L'entità Tipologia,identificata univocamente da un IdTipologia è descritta da due attributi quali Ora Inizio ed Ora Fine , è un elenco di tutte le possibili tipologie di permesso.
Essa è legata a “Permesso” tramite l'associazione “di” con cardinalità (0,n) in quanto ad una certa tipologia possono appartenere più permessi. Permesso invece è associato a Tipologia con cardinalità (1,1) perchè un certo permesso può appartenere ad un sola tipologia.

Accesso/Rifuto:



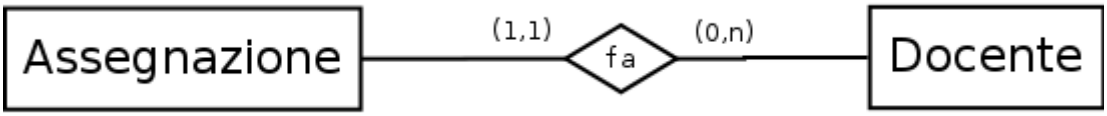
Queste due associazione mettono in evidenza il concetto di Accesso ad un Locale e di Rifiuto : per accesso intendiamo l'atto con cui un utente riesce con successo , utilizzando la sua tessera magnetica, ad accedere ad un certo locale; per rifiuto si intende il momento in cui il sistema rifiuta l'acceso a quel locale da parte di un certo utente, in quanto magari la tipologia di permesso non lo ammette.
Con una Tessera Magnetica un certo utente può accedere ad un solo locale in una certa data e in una certa ora, di modo da evitare di poter inserire un accesso contemporaneo di un utente in più locali differenti . A tale scopo la chiave di Accesso è composta da data,ora e attraverso identificazione esterna dall'IdTessera.
Analogo discorso vale per l'entità Rifiuto.

Associazione Laureando/Assegnati



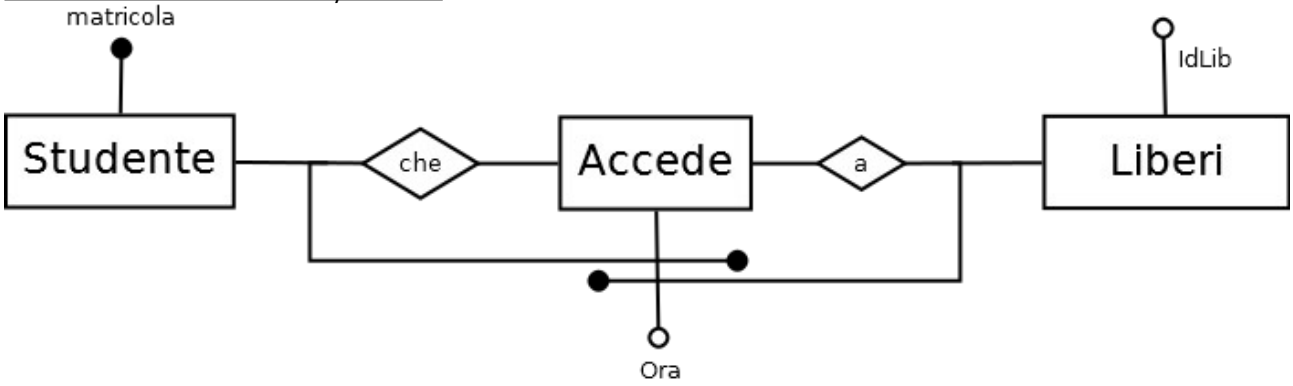
Tale associazione descrive l'assegnazione di un posto di lavoro ad un laureando. Essa è mensile, ovvero in un certo mese un posto di lavoro assegnabile è dato ad un solo laureando, inoltre un laureando in un certo mese può ottenere l'assegnazione di un solo posto. Tutto ciò previa approvazione di un docente (segue).

Associazione Assegnazione/Docente



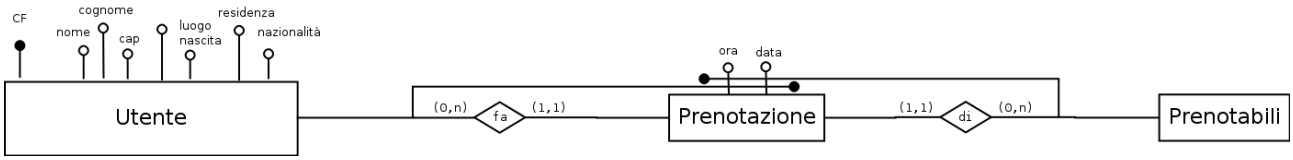
Come sopracitato un'assegnazione di un posto di lavoro ad un laureando deve essere fatta da un docente. Un docente può sottoscrivere da 0 a n assegnazioni, ma un'assegnazione è fatta chiaramente da un solo docente.

Associazione Sudente/Liberi



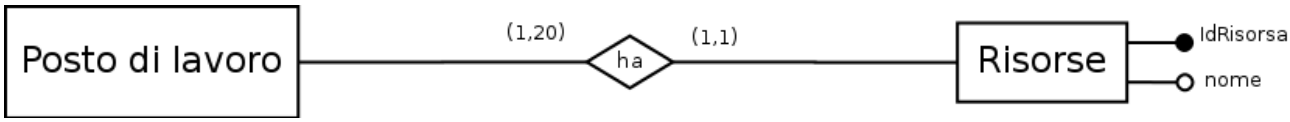
I posti di lavoro liberi sono accessibili da tutti gli studenti, senza bisogno di alcuna prenotazione, l'unica cosa richiesta è registrare l'utilizzo di essi. Uno studente può accedere da 0 a n posti di lavoro liberi, e un posto può essere occupato da altrettanti studenti. Ovviamente in una certa ora un posto di lavoro può essere occupato da un solo studente e, viceversa, uno studente può occupare un solo posto.

Associazione Utente/Prenotabili



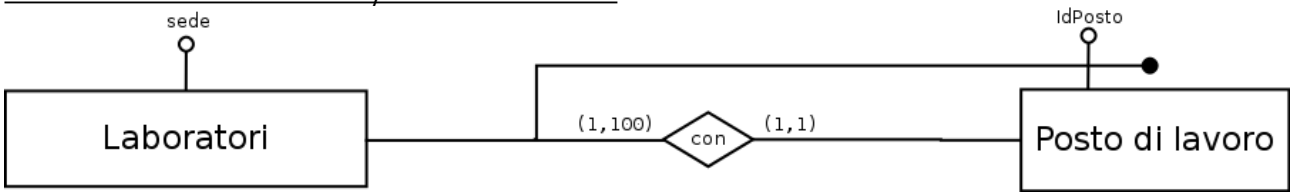
Un posto prenotabile è un posto di lavoro che può essere assegnato giornalmente ad uno studente o ad un docente titolare di un corso. Abbiamo usato l'entità Utente per evitare conflitti di prenotazioni, come sovrapposizioni di orari, sarà un trigger a fare in modo che la prenotazione non venga eseguita in caso sia un non docente o un docente non titolare a richiedere la prenotazione stessa.
Inoltre la prenotazione è intesa ad ore :se per esempio uno studente volesse prenotare un certo posto dalle 14:00 alle 17:00 allora dovrà inserire tre prenotazioni : una per l'ora 14:00-15:00, una per 15:00-16:00 e una per 16:00-17:00. Abbiamo optato per questa strada poiché risultava impossibile gestire le prenotazione per intervalli di tempo, ci sarebbe stato il rischio di poter prenotare lo stesso posto di due intervalli temporali diversi ma sovrapposti.
Secondo la nostra soluzione un posto può essere prenotato in una certa data e ora da un solo utente, mentre un utente in una certa data e ora può prenotare un solo posto.

Assegnazione Posto di lavoro/Risorse



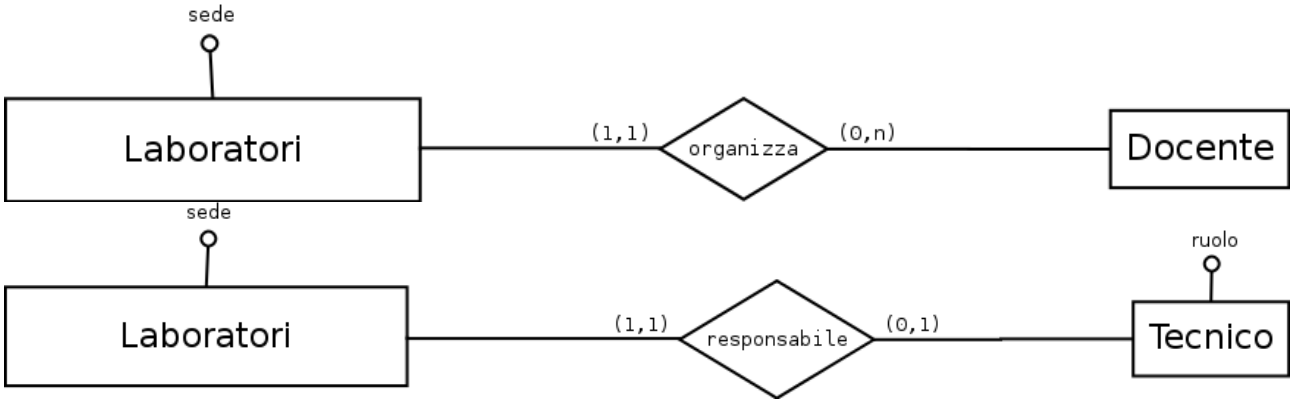
I posti di lavoro possiedono delle risorse, da un minimo di una ad un massimo di venti, che possono essere usate da chi accede a tale posto. Per esempio in un laboratorio di informatica un posto di lavoro sarà dotato di un computer con installati certi software applicativi; per un laboratorio di biologia vi sarà eventualmente un microscopio, alcuni vetrini e strumenti ecc.
Una risorsa ovviamente appartiene ad un solo posto di lavoro, anche se sarà poi possibile modificare a quale appartiene con un'operazione di update specificata in seguito.

Associazione Laboratori/Posto di lavoro



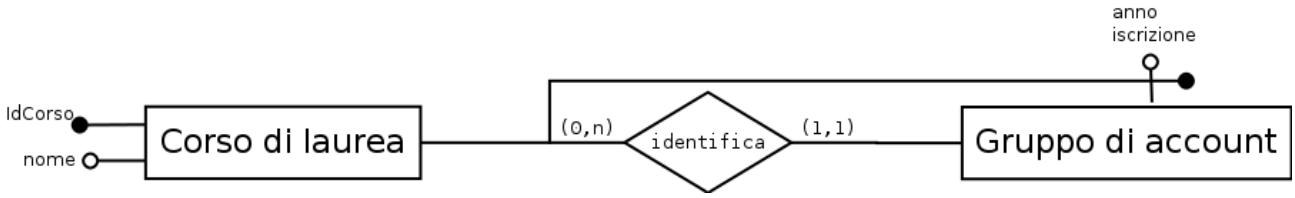
Tale associazione descrive il legame tra un laboratorio e un posto di lavoro. Un posto di lavoro è identificato da un codice univoco all'interno del laboratorio a cui appartiene. In un laboratorio vi sono al massimo cento posti di lavoro e un posto di lavoro non può che appartenere ad un solo laboratorio. Il numero di posti di lavoro non coincide con la capacità di un locale : in un locale infatti vi può essere una capacità anche molto maggiore del numero dei posti di lavoro.

Associazioni Laboratori/Docente e Laboratori/Tecnico



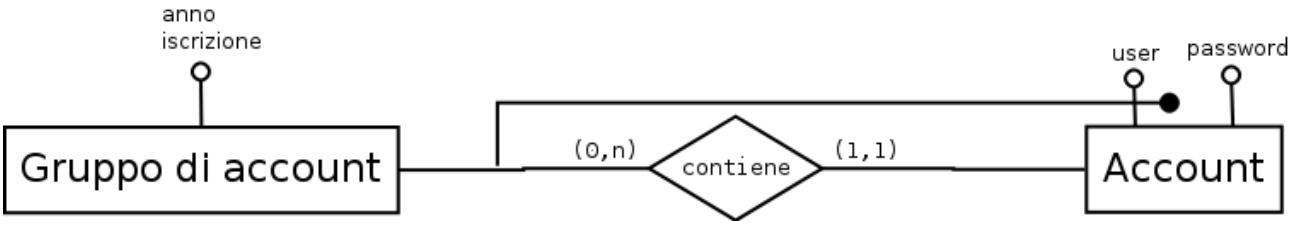
Per ogni laboratorio vi sono due persone particolari e necessarie : un docente che organizzi tale laboratorio e un tecnico che ne sia responsabile. Un docente può organizzare quanti laboratori vuole, ma un certo laboratorio può essere organizzato da un unico docente. Per un tecnico invece è diverso : egli può essere responsabile di un solo laboratorio, e il laboratorio è sotto la responsabilità di un unico t.

Associazione Corso di Laurea/Gruppo di Account



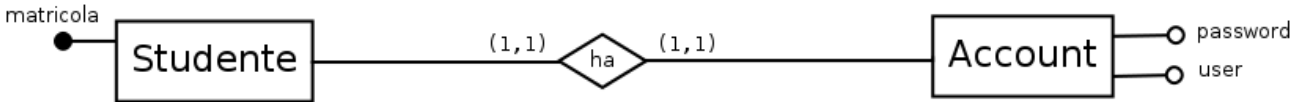
Tale associazione descrive il legame tra il gruppo di account e il corso di laurea. Infatti un gruppo di account è un'entità debole rispetto al corso di laurea e ha in chiave l'id del corso e l'anno di iscrizione. In tal modo per ogni corso e un anno di iscrizione vi sarà un ed un solo gruppo di account.

Associazione Gruppo di account/Account



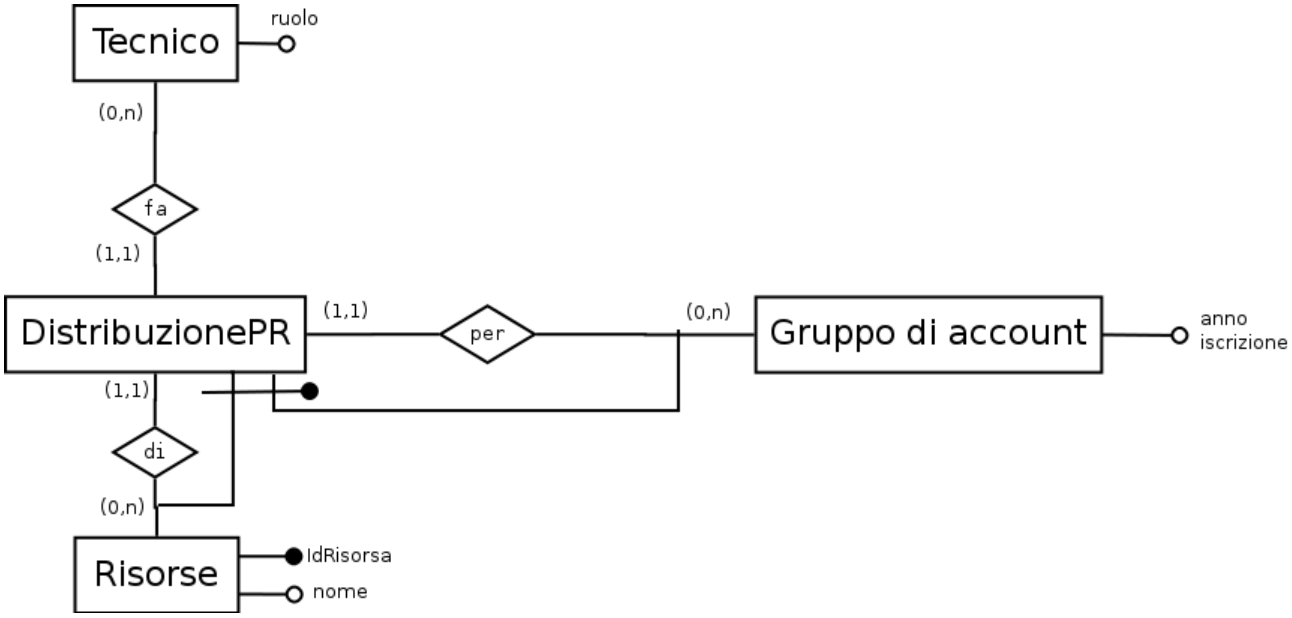
Per ogni gruppo di account, identificati dall'anno di iscrizione e del corso di laurea, vi sono da 0 a n account. Gli account appartengono ovviamente ad un solo gruppo e sono identificati, oltre che dall'anno di iscrizione e dall'id del corso, da un User univoco e hanno come attributo una password necessaria.

Associazione Studente/Account



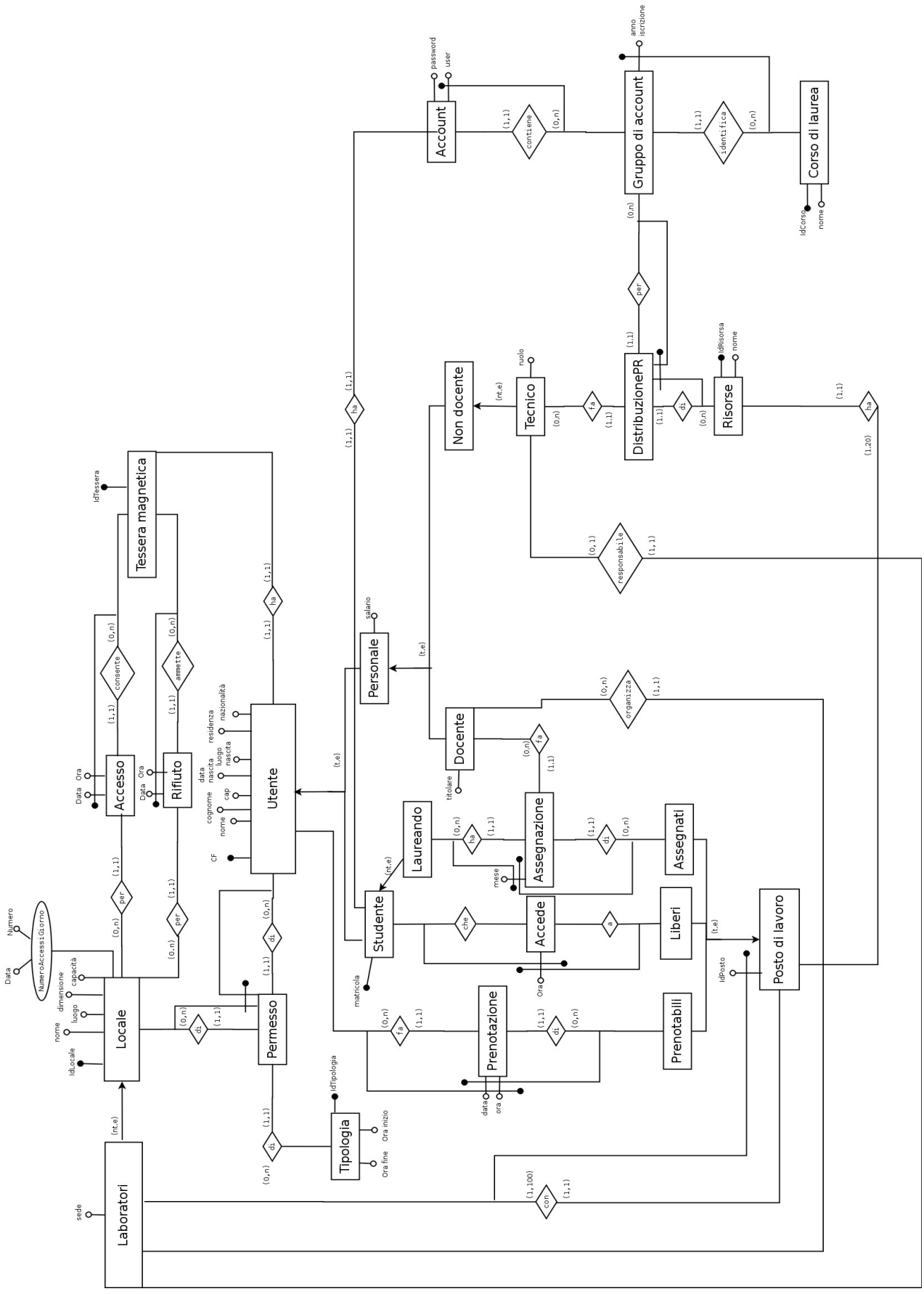
Ogni studente universitario ha un solo account, e tale account appartiene solo a lui. Ovviamente gli sarà assegnato tale account in base al corso di laurea che frequenta e all'anno in cui vi si è iscritto.

Associazione Tecnico/Risorse/Gruppo di Account



Ad ogni account sono assegnate alcune risorse. Noi abbiamo gestito tale vincolo spostandolo sul gruppo di account, così da assegnare a tutti gli account dello stesso gruppo le stesse risorse. Tale assegnazione è fatta da uno o più tecnici, che però devono essere System Administrator (vincolo gestito da un trigger). Essi possono associare più risorse a più gruppi. Una stessa risorsa può essere assegnata a più gruppi, infatti se per esempio la risorsa è un software applicativo, allora esso può essere reso disponibile per più gruppi di account.

5 - Progetto concettuale completo



6-Progetto logico-schema relazionale

Utente(CF,Nome,Cognome,DataN,LuogoN,Residenza,CAP,Nazionalità,CodTessera)
AK: CodTessera
Studente (CF,Matr,CodC,AnnoI,User)
AK:Matr
FK:CF reference Utente
FK: CodC,AnnoI,User reference Account
Laureando (CF)
FK: CF reference Studente
Personale (CF,IdP,Salario)
AK: IdP
FK:CF reference Utente
Docente(CF, titolare)
FK: CF reference Personale
NonDocente(CF)
FK: CF reference Personale
Tecnico(CF, Ruolo)
FK: CF reference NonDocente
Assegnato (IdPosto,IdLoc)
FK: IdPosto,IdLoc reference PostoDiLavoro
Libero(IdPosto,IdLoc)
FK: IdPosto,IdLoc reference PostoDiLavoro
Prenotabile(IdPosto,IdLoc)
FK: IdPosto,IdLoc reference PostoDiLavoro
Assegnazione (Mese,CFL,IdPosto,IdLoc, CFD)
AK: Mese,IdPosto,IdLoc
FK: CFL reference Laureando
FK: CFD reference Docente
FK: IdPosto,IdLoc reference Assegnato
Prenotazione (Data, Ora, IdPosto,IdLoc, CF)
AK : CF,Data,Ora
FK: CF reference Utente
FK: IdPosto,IdLoc reference Prenotabile
Accede (Ora,CF,IdLib,IdLoc)
AK: Ora,IdLib,IdLoc
FK:CF reference Studente
FK:IdLib,IdLoc reference Libero
Risorsa (IdRis, Nome, IdPosto,IdLoc)
FK: IdPosto,IdLoc reference PostoDiLavoro
DistribuzionePR (IdCDL, AnnoIscrizione, IdRis,CF)
FK: IdCDL,AnnoIscrizione reference CorsoDiLaurea
FK: IdRis reference Risorsa
FK: CF reference Tecnico
Permesso (CF,IdLoc,IdT)
FK: CF reference Utente
FK: IdLoc reference Locale
FK: IdT reference Tipologia
Tipologia (IdT, OraInizio,OraFine,Giorno)
Locale (IdLoc, Nome, Luogo, Dimensioni, Capacità)
NumAccessi(IdLoc,Data,Numero)

FK : IdLoc reference Locale

PostoDiLavoro (IdPosto, IdLoc)

FK: IdLoc reference Laboratorio

Laboratorio (IdLoc,Sede,CFT,CFD)

FK:IdLoc reference Locale

FK: CFT reference Tecnico

FK: CFD reference Docente

Rifiuto (Data, Ora, CodTessera, IdLoc)

FK: CodTessera reference Utente

FK: IdLoc refrence Locale

Accesso(Data, Ora, CodTessera, IdLoc)

FK: CodTessera reference Utente

FK: IdLoc refrence Locale

Account (IdCDL, AnnoIscrizione,User,Password)

FK: IdCDL,AnnoIscrizione reference GruppoDiAccount

GruppoDiAccount (IdCDL, AnnoIscrizione)

FK: IdCdL reference CorsoDiLaurea

CorsoDiLaurea (IdCDL, Nome)

7-SQL creazione tabelle e popolamento

```
CREATE TABLE TIPOLOGIA
(
      IdT         INTEGER NOT NULL,
      OraInizio   TIME NOT NULL,
      OraFine           TIME NOT NULL,

      PRIMARY KEY (IdT),
      CHECK(OraFine > OraInizio)
);

CREATE TABLE CORSOLAUREA
(
      IdCDL INTEGER NOT NULL,
      Nome  CHAR(50)  ,

      PRIMARY KEY (IdCDL)
);

CREATE TABLE GRUPPOACCOUNT
(
      IdCDL       INTEGER NOT NULL,
      AnnoIscrizione   INTEGER NOT NULL,

      PRIMARY KEY (IdCDL, AnnoIscrizione),
      FOREIGN KEY (IdCDL) REFERENCES CORSOLAUREA (IdCDL)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE ACCOUNT
(
      IdCDL       INTEGER NOT NULL,
      AnnoIscrizione   INTEGER NOT NULL,
      NomeUser    CHAR(50) NOT NULL UNIQUE,
      Password    CHAR(50) NOT NULL,

      PRIMARY KEY (IdCDL, AnnoIscrizione,NomeUser),
      FOREIGN KEY (IdCDL, AnnoIscrizione) REFERENCES GRUPPOACCOUNT
(IdCDL, AnnoIscrizione)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE UTENTE
(
      CF          CHAR(16) NOT NULL,
      Nome        CHAR(50),
      Cognome           CHAR(50),
      DataN             DATE,
      LuogoN            CHAR(50),
      Residenza   CHAR(50),
      CAP         INTEGER,
      Nazionalita       CHAR(50),
      CodTessera INTEGER NOT NULL,
```

```
      PRIMARY KEY (CF),
      UNIQUE (CodTessera)
);

CREATE TABLE STUDENTE
(
      CF          CHAR(16) NOT NULL,
      Matr        INTEGER NOT NULL,
      IdCDL       INTEGER,
      AnnoI       INTEGER,
      NomeUser    CHAR(50),

      PRIMARY KEY (CF),
      UNIQUE (Matr),
      FOREIGN KEY (CF) REFERENCES UTENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (IdCDL,AnnoI,NomeUser) REFERENCES
ACCOUNT(IdCDL,AnnoIscrizione,NomeUser)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE LAUREANDO
(
      CF    CHAR(16) NOT NULL,

      PRIMARY KEY (CF),
      FOREIGN KEY (CF) REFERENCES STUDENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE PERSONALE
(
      CF    CHAR(16) NOT NULL,
      Salario     INTEGER ,
      IdP   INTEGER NOT NULL,

      PRIMARY KEY (CF),
      UNIQUE (IdP),
      FOREIGN KEY (CF) REFERENCES UTENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE DOCENTE
(
      CF          CHAR(16) NOT NULL,
      TITOLARE    BOOLEAN NOT NULL,

      PRIMARY KEY (CF),
      FOREIGN KEY (CF) REFERENCES PERSONALE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE NONDOCENTE
(
      CF    CHAR(16) NOT NULL,
```

```
      PRIMARY KEY (CF),
      FOREIGN KEY (CF) REFERENCES PERSONALE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE TECNICO
(
      CF    CHAR(16) NOT NULL,
      Ruolo CHAR(50),

      PRIMARY KEY (CF),
      FOREIGN KEY (CF) REFERENCES NONDOCENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE LOCALE
(
      IdLoc       INTEGER NOT NULL,
      Nome        CHAR(50),
      Luogo       CHAR(50),
      Dimensioni  DECIMAL(6,4),
      Capacita    INTEGER ,

      PRIMARY KEY (IdLoc)
);

CREATE TABLE NUMACCESSI
(
      IdLoc INTEGER NOT NULL,
      Numero      INTEGER,
      Data  DATE  NOT NULL,

      PRIMARY KEY (IdLoc,Data),
      FOREIGN KEY (IdLoc) REFERENCES LOCALE (IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE LABORATORIO
(
      IdLoc       INTEGER NOT NULL,
      Sede        CHAR(50),
      CFT   CHAR(16) NOT NULL,
      CFD   CHAR(16) NOT NULL,

      PRIMARY KEY (IdLoc),
      FOREIGN KEY (IdLoc) REFERENCES LOCALE(IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CFT) REFERENCES TECNICO(CF)
            ON DELETE SET NULL ON UPDATE CASCADE,
      FOREIGN KEY (CFD) REFERENCES DOCENTE(CF)
            ON DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE PDLAVORO
(
```

```
      IdPosto    INTEGER NOT NULL UNIQUE,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (IdPosto, IdLoc),
      FOREIGN KEY (IdLoc) REFERENCES LABORATORIO(IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE ASSEGNATO
(
      IdPosto    INTEGER NOT NULL,
      IdLoc      INTEGER NOT NULL,

      PRIMARY KEY (IdPosto,IdLoc),
      FOREIGN KEY (IdPosto,IdLoc)REFERENCES PDLAVORO(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE LIBERO
(
      IdPosto    INTEGER NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (IdPosto,IdLoc),
      FOREIGN KEY (IdPosto,IdLoc)REFERENCES PDLAVORO(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE PRENOTABILE
(
      IdPosto    INTEGER NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (IdPosto,IdLoc),
      FOREIGN KEY (IdPosto,IdLoc)REFERENCES PDLAVORO(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE ASSEGNAZIONE
(
      Mese  INTEGER NOT NULL,
      CFL   CHAR(16) NOT NULL,
      CFD   CHAR(16) NOT NULL,
      IdPosto    INTEGER NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (Mese,CFL),
      UNIQUE  (Mese,IdPosto,IdLoc),
      FOREIGN KEY (IdPosto,IdLoc)REFERENCES ASSEGNATO(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CFD) REFERENCES DOCENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CFL) REFERENCES LAUREANDO(CF)
            ON DELETE CASCADE ON UPDATE CASCADE,
      CHECK (Mese>0 and mese<=12)
);
```

```
CREATE TABLE PRENOTAZIONE
(
      Data  DATE NOT NULL,
      Ora   TIME NOT NULL,
      IdPosto    INTEGER NOT NULL,
      CF    CHAR(16) NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (Data,Ora,IdPosto,IdLoc),
      UNIQUE  (CF,Data,Ora),
      FOREIGN KEY (CF) REFERENCES UTENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (IdPosto,IdLoc)REFERENCES PRENOTABILE(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE ACCEDE
(
      IdLib INTEGER NOT NULL,
      CF    CHAR(16) NOT NULL,
      Ora   TIME NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (Ora,CF),
      UNIQUE (Ora,IdLib,IdLoc),
      FOREIGN KEY (CF) REFERENCES STUDENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (IdLib,IdLoc)REFERENCES LIBERO(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE RISORSA
(
      IdRis INTEGER NOT NULL,
      IdPosto    INTEGER ,
      Nome  CHAR(50),
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (IdRis),
      FOREIGN KEY (IdPosto,IdLoc)REFERENCES PDLAVORO(IdPosto,IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE DISTRIBUZIONEPR
(
      IdCDL      INTEGER NOT NULL,
      AnnoIscrizione  INTEGER NOT NULL,
      IdRis      INTEGER NOT NULL,
      CF         CHAR(16) NOT NULL,

      PRIMARY KEY (IdCDL, AnnoIscrizione, IdRis),
      FOREIGN KEY (IdCDL)REFERENCES CORSOLAUREA(IdCDL)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (IdRis)REFERENCES RISORSA(IdRis)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CF) REFERENCES TECNICO(CF)
```

```
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE PERMESSO
(
      CF    CHAR(16) NOT NULL,
      IdT   INTEGER NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (CF, IdLoc),
      FOREIGN KEY (IdLoc) REFERENCES LOCALE(IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CF) REFERENCES UTENTE(CF)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (IdT) REFERENCES TIPOLOGIA(IdT)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE RIFIUTO
(
      Data  DATE NOT NULL,
      Ora   TIME NOT NULL,
      CodTessera INTEGER NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (Data,Ora,CodTessera),
      FOREIGN KEY (IdLoc) REFERENCES LOCALE(IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CodTessera) REFERENCES UTENTE(CodTessera)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE ACCESSO
(     Data  DATE NOT NULL,
      Ora   TIME NOT NULL,
      CodTessera INTEGER NOT NULL,
      IdLoc INTEGER NOT NULL,

      PRIMARY KEY (Data,Ora,CodTessera),
      FOREIGN KEY (IdLoc) REFERENCES LOCALE(IdLoc)
            ON DELETE CASCADE ON UPDATE CASCADE,
      FOREIGN KEY (CodTessera) REFERENCES UTENTE(CodTessera)
            ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
INSERT INTO Utente
VALUES('ASDFNH92L010K','Mario','Rossi','24/12/1992','Bondeno','strada
zocca 3',46022,'italiano',21340);
INSERT INTO Utente
VALUES('SDFRTY45L871H','Pietro','Grandi','02/02/1992','Sassuolo','strada
dei caduti 1',45234,'italiano',22354);
INSERT INTO Utente
VALUES('ERTYUI23L543D','Sara','Cortese','03/06/1994','Mantova','strada
valla del oca 12',45678,'italiano',34560);
INSERT INTO Utente
VALUES('QWEASD12G234C','Giovanna','Trombelli','06/07/1991','Maranello','v
ia corso garibaldi 12',34562,'italiano',12345);
INSERT INTO Utente
VALUES('POIUYT23S543S','Mirko','Movida','07/07/1969','Bancole','via non
lo so 13',54678,'italiano',23675);
INSERT INTO Utente
VALUES('ZXCVBN12D563A','Valentino','Rossi','09/08/1953','Modena','via
Garibaldi 10',34231,'italiano',12002);
INSERT INTO Utente
VALUES('ZAFCDF16D513F','Valentino','Marconi','29/08/1970','Modena','via
Garibaldi 11',34231,'italiano',12456);
INSERT INTO Utente
VALUES('GHCXBA45D163A','Mario','Mosconi','19/10/1963','Modena','via
Verucchi 10',34231,'italiano',15672);
INSERT INTO Utente
VALUES('LKJHGF12A094D','Pino','Lavatrice','01/01/1971','Modena','via
Trento-Trieste 32',34231,'italiano',23455);
INSERT INTO Utente
VALUES('KJHDSA92T242S','Alberto','Rossi','30/10/1980','Modena','via
Vignolese 123',34231,'italiano',12356);

INSERT INTO CorsoLaurea VALUES(100,'Informatica');
INSERT INTO CorsoLaurea VALUES(210,'Ing. Meccanica');
INSERT INTO CorsoLaurea VALUES(110,'Matematica');
INSERT INTO CorsoLaurea VALUES(112,'Fisica');
INSERT INTO CorsoLaurea VALUES(310,'Economia');

INSERT INTO GruppoAccount VALUES(100,2011);
INSERT INTO GruppoAccount VALUES(100,2012);
INSERT INTO GruppoAccount VALUES(310,2013);
INSERT INTO GruppoAccount VALUES(110,2010);
INSERT INTO GruppoAccount VALUES(112,2014);

INSERT INTO Account VALUES(110,2010,'ciao','mamma');
INSERT INTO Account VALUES(110,2010,'sempo','c1992');
INSERT INTO Account VALUES(112,2014,'davide','dav23121992');
INSERT INTO Account VALUES(310,2013,'mirko','ssdf123');

INSERT INTO Studente VALUES('ASDFNH92L010K',71498,110,2010,'ciao');
INSERT INTO Studente VALUES('SDFRTY45L871H',34566,110,2010,'sempo');
INSERT INTO Studente VALUES('ERTYUI23L543D',87654,112,2014,'davide');
INSERT INTO Studente VALUES('QWEASD12G234C',72298,310,2013,'mirko');

INSERT INTO Laureando VALUES('ASDFNH92L010K');
INSERT INTO Laureando VALUES('ERTYUI23L543D');
```

```
INSERT INTO Personale VALUES('POIUYT23S543S',32000,1);
INSERT INTO Personale VALUES('ZXCVBN12D563A',45000,2);
INSERT INTO Personale VALUES('ZAFCDF16D513F',23000,3);
INSERT INTO Personale VALUES('GHCXBA45D163A',31000,4);
INSERT INTO Personale VALUES('LKJHGF12A094D',23000,5);
INSERT INTO Personale VALUES('KJHDSA92T242S',16000,6);

INSERT INTO Docente VALUES('ZXCVBN12D563A',true);
INSERT INTO Docente VALUES('GHCXBA45D163A',true);
INSERT INTO Docente VALUES('POIUYT23S543S',false);

INSERT INTO NonDocente VALUES('ZAFCDF16D513F');
INSERT INTO NonDocente VALUES('LKJHGF12A094D');
INSERT INTO NonDocente VALUES('KJHDSA92T242S');

INSERT INTO Tecnico VALUES('KJHDSA92T242S','System Administrator');
INSERT INTO Tecnico VALUES('LKJHGF12A094D','System Administrator');
INSERT INTO Tecnico VALUES('ZAFCDF16D513F','Responsabile materiale
didattico');

INSERT INTO Locale VALUES(001,'Laboratorio base','Edificio
Matematica',80,60);
INSERT INTO Locale VALUES(002,'Sala disegno','Ingegneria',90,65);
INSERT INTO Locale VALUES(003,'Lab. chimica','Farmacia',60,70);
INSERT INTO Locale VALUES(004,'Aula V','Matematica',50,100);
INSERT INTO Locale VALUES(005,'Laboratorio fisica
nucleare','Fisica',90,50);
INSERT INTO Locale VALUES(006,'Laboratorio biologia','Biologia',75,60);
INSERT INTO Locale VALUES(007,'Laboratorio
meccanica','Ingengeria',99,50);
INSERT INTO Locale VALUES(008,'BSI','Campus Unimore',98,300);
INSERT INTO Locale VALUES(009,'CUS','Campus Unimore',95,200);
INSERT INTO Locale VALUES(010,'Tito Speri','Biotecnologia',97,350);

INSERT INTO Laboratorio VALUES(001,'Edificio
Matematica','KJHDSA92T242S','ZXCVBN12D563A');
INSERT INTO Laboratorio
VALUES(003,'Farmacia','LKJHGF12A094D','POIUYT23S543S');
INSERT INTO Laboratorio
VALUES(006,'Biologia','ZAFCDF16D513F','ZXCVBN12D563A');

INSERT INTO PDLavoro VALUES(0001,001);
INSERT INTO PDLavoro VALUES(0002,006);
INSERT INTO PDLavoro VALUES(0003,003);
INSERT INTO PDLavoro VALUES(0004,001);
INSERT INTO PDLavoro VALUES(0005,003);
INSERT INTO PDLavoro VALUES(0006,006);
INSERT INTO PDLavoro VALUES(0007,001);
INSERT INTO PDLavoro VALUES(0008,003);
INSERT INTO PDLavoro VALUES(0009,006);
INSERT INTO PDLavoro VALUES(0010,001);

INSERT INTO Assegnato VALUES(0001,001);
INSERT INTO Assegnato VALUES(0009,006);
INSERT INTO Assegnato VALUES(0007,001);
```

```
INSERT INTO Libero VALUES(0002,006);
INSERT INTO Libero VALUES(0008,003);

INSERT INTO Prenotabile VALUES(0003,003);
INSERT INTO Prenotabile VALUES(0004,001);
INSERT INTO Prenotabile VALUES(0005,003);
INSERT INTO Prenotabile VALUES(0006,006);
INSERT INTO Prenotabile VALUES(0010,001);

INSERT INTO Assegnazione
VALUES(02,'ASDFNH92L010K','ZXCVBN12D563A',0001,001);
INSERT INTO Assegnazione
VALUES(03,'ASDFNH92L010K','ZXCVBN12D563A',0001,001);
INSERT INTO Assegnazione
VALUES(04,'ERTYUI23L543D','POIUYT23S543S',0007,001);
INSERT INTO Assegnazione
VALUES(07,'ASDFNH92L010K','GHCXBA45D163A',0009,006);
INSERT INTO Assegnazione
VALUES(08,'ASDFNH92L010K','GHCXBA45D163A',0009,006);
INSERT INTO Assegnazione
VALUES(10,'ERTYUI23L543D','POIUYT23S543S',0007,001);
INSERT INTO Assegnazione
VALUES(11,'ERTYUI23L543D','GHCXBA45D163A',0009,006);
INSERT INTO Assegnazione
VALUES(12,'ASDFNH92L010K','GHCXBA45D163A',0001,001);
INSERT INTO Assegnazione
VALUES(12,'ERTYUI23L543D','POIUYT23S543S',0007,001);

INSERT INTO Prenotazione
VALUES('24/01/2013','14:00',0003,'ASDFNH92L010K',003);
INSERT INTO Prenotazione
VALUES('24/01/2013','14:00',0004,'ERTYUI23L543D',001);
INSERT INTO Prenotazione
VALUES('30/01/2013','09:00',0005,'GHCXBA45D163A',003);
INSERT INTO Prenotazione
VALUES('01/02/2013','10:00',0003,'QWEASD12G234C',003);
INSERT INTO Prenotazione
VALUES('09/02/2013','11:00',0006,'ASDFNH92L010K',006);
INSERT INTO Prenotazione
VALUES('10/02/2013','14:00',0006,'ERTYUI23L543D',006);
INSERT INTO Prenotazione
VALUES('16/03/2013','18:00',0003,'GHCXBA45D163A',003);
INSERT INTO Prenotazione
VALUES('01/04/2013','17:00',0010,'ASDFNH92L010K',001);
INSERT INTO Prenotazione
VALUES('09/04/2013','13:00',0005,'SDFRTY45L871H',003);
```

Esempio di tupla errata : tale inserimento non è accettato dal trigger poiché il CF usato appartiene ad un tecnico, il quale non ha il permesso di prenotare posti di lavoro.

```
INSERT INTO Prenotazione
VALUES('14/02/2013','16:00',0010,'ZAFCDF16D513F',001);

INSERT INTO Accede VALUES(0002,'ASDFNH92L010K','14:00',006);
INSERT INTO Accede VALUES(0008,'SDFRTY45L871H','16:00',003);
```

```
INSERT INTO Accede VALUES(0002,'QWEASD12G234C','17:00',006);
INSERT INTO Accede VALUES(0008,'ERTYUI23L543D','11:00',003);

INSERT INTO Risorsa VALUES(1,0001,'Windows Office',001);
INSERT INTO Risorsa VALUES(2,0004,'Stampante',001);
INSERT INTO Risorsa VALUES(3,0006,'shell',006);
INSERT INTO Risorsa VALUES(4,0001,'Campo minato',001);
INSERT INTO Risorsa VALUES(5,0006,'Scanner',006);
INSERT INTO Risorsa VALUES(6,0002,'Beuta',006);
INSERT INTO Risorsa VALUES(7,0001,'Microscopio',001);
INSERT INTO Risorsa VALUES(8,0009,'Pipetta',006);
INSERT INTO Risorsa VALUES(9,0001,'Dinamometro',001);
INSERT INTO Risorsa VALUES(10,0010,'Vetrino',001);

INSERT INTO DistribuzionePR VALUES(100,2011,1,'KJHDSA92T242S');
INSERT INTO DistribuzionePR VALUES(112,2014,2,'LKJHGF12A094D');
INSERT INTO DistribuzionePR VALUES(112,2014,1,'LKJHGF12A094D');
INSERT INTO DistribuzionePR VALUES(112,2014,9,'KJHDSA92T242S');
INSERT INTO DistribuzionePR VALUES(100,2011,2,'LKJHGF12A094D');
INSERT INTO DistribuzionePR VALUES(110,2010,1,'LKJHGF12A094D');
INSERT INTO DistribuzionePR VALUES(310,2013,9,'LKJHGF12A094D');
INSERT INTO DistribuzionePR VALUES(110,2010,5,'KJHDSA92T242S');
INSERT INTO DistribuzionePR VALUES(310,2013,7,'KJHDSA92T242S');
```

Esempio di tupla errata : tale inserimento non è accettato dal trigger perchè il CF inserito appartiene ad un tecnico che non è System Administrator.

```
INSERT INTO DistribuzionePR VALUES(100,2011,10,'ZAFCDF16D513F');

INSERT INTO Tipologia VALUES(100,'14:00','16:00');
INSERT INTO Tipologia VALUES(101,'8:00','19:00');
INSERT INTO Tipologia VALUES(102,'10:00','12:00');
INSERT INTO Tipologia VALUES(103,'9:00','20:00');
INSERT INTO Tipologia VALUES(104,'14:00','18:00');
INSERT INTO Tipologia VALUES(105,'8:30','12:00');

INSERT INTO Permesso VALUES('ASDFNH92L010K',100,001);
INSERT INTO Permesso VALUES('SDFRTY45L871H',102,008);
INSERT INTO Permesso VALUES('ZXCVBN12D563A',105,004);
INSERT INTO Permesso VALUES('ZXCVBN12D563A',104,008);
INSERT INTO Permesso VALUES('ASDFNH92L010K',103,009);
INSERT INTO Permesso VALUES('ERTYUI23L543D',102,001);
INSERT INTO Permesso VALUES('KJHDSA92T242S',101,003);
INSERT INTO Permesso VALUES('POIUYT23S543S',100,001);

INSERT INTO Accesso VALUES('24/01/2013','15:00',21340,001);
INSERT INTO Accesso VALUES('02/02/2013','10:00',22354,008);
INSERT INTO Accesso VALUES('10/02/2013','11:00',12002,004);
INSERT INTO Accesso VALUES('15/02/2013','15:00',12002,008);
INSERT INTO Accesso VALUES('01/03/2013','19:00',21340,009);
```

Esempio di tuple errate : Tali tuple non sono accettate dal trigger perchè gli utenti non hanno il permesso di accedere a tali locali in tale ora della giornata. Queste tuple non vengono inserite in ACCESSO ma in RIFIUTO, per cui il trigger non bloccherà l'inserimento.

```
INSERT INTO Accesso VALUES('15/03/2013','09:00',34560,001);
INSERT INTO Accesso VALUES('04/04/2013','07:00',12356,003);
INSERT INTO Accesso VALUES('10/04/2013','20:00',23675,001);
INSERT INTO Accesso VALUES ('10/04/2013','21:00',23675,001);
INSERT INTO Accesso VALUES ('10/04/2013','22:00',23675,001);
```

8- Operazioni

1. Dato un locale elencare tutti gli utenti che hanno il permesso di accedere al locale con i relativi giorni e intervalli temporali di accesso.
2. Selezionare nome,cognome degli utenti che hanno prenotato più di 2 posti un certo mese.
3. Visualizzare chi ha avuto più di 3 rifiuti in uno stesso giorno lo stesso giorno.
4. Visualizzare le assegnazioni di posti di lavoro ai laureandi nel mese M
5. Visualizzare tutte le risorse assegnate ad un gruppo di account X e non all'Y.
6. Selezionare un locale che non ha avuto rifiuti.
7. Selezionare un locale che abbia almeno un rifiuto
8. Selezionare I gruppi di account con risorsa “risorsa” assegnata dal tecnico “tecnico”
9. Visualizzare gli intervalli orari in cui è permesso accedere al laboratorio “Laboratorio”;
10. Eliminare una prenotazione fatta da un utente;
11. Eliminare una Risorsa;
12. Eliminare un utente;
13. Aggiornare un intervallo della tipologia di permesso;
14. Aggiornare dati anagrafici utente;
15. Modificare posto di lavoro alle risorse;

9 - SQL operazioni

1. Dato un locale elencare tutti gli utenti che hanno il permesso di accedere al locale con i relativi giorni e intervalli temporali di accesso.

```
SELECT u.Nome,u.Cognome,t.OraInizio, t.OraFine
FROM UTENTE AS U,PERMESSO AS P, TIPOLOGIA AS T
WHERE P.CF = U.CF AND T.IdT = P.IdT AND P.IdLoc = locCercato
```

2. Selezionare nome,cognome degli utenti che hanno prenotato più di 2 posti un certo mese.

```
SELECT U.nome, U.cognome
FROM UTENTE AS U
WHERE  EXISTS (SELECT COUNT(*)
                  FROM PRENOTAZIONE AS P
                  WHERE U.CF = P.CF AND Extract(month from P.Data) =
CertoMese
                  GROUP BY P.CF
                  HAVING COUNT(*)>=2)
```

3. Visualizzare chi ha avuto più di 3 rifiuti in uno stesso giorno lo stesso giorno.

```
SELECT U.nome, R.data, R.idloc
FROM UTENTE AS U, RIFIUTO AS R
WHERE U.CodTessera = R.CodTessera AND R.Data = 'CertaData' AND
R.IdLoc =  certoLocale
GROUP BY U.nome, R.data, R.idloc
HAVING (COUNT(*)>=3)
```

4. Visualizzare le assegnazioni di posti di lavoro ai laureandi nel mese M

```
SELECT L.cf
FROM LAUREANDO AS L, ASSEGNAZIONE AS A
WHERE L.CF = A.CFL AND A.Mese =M
```

5. Visualizzare tutte le risorse assegnate ad un gruppo di account X e non all'Y.

```
SELECT IdRis
FROM DISTRIBUZIONEPR
WHERE IdCDL = IdX AND AnnoIscrizione = AnnoX AND IdRis
           NOT IN (   SELECT IdRis
                      FROM DISTRIBUZIONEPR
                      WHERE IdCDL=IdY AND AnnoIscrizione=AnnoY)
```

6. Selezionare un locale che non ha avuto rifiuti.

```
SELECT IdLoc
FROM LOCALE
WHERE IdLoc NOT IN(   SELECT IdLoc
                      FROM RIFIUTO)
```

7. Selezionare un locale che abbia almeno un rifiuto

```
SELECT *
FROM LOCALE as l0
WHERE NOT EXISTS(SELECT *
           FROM LOCALE as l1
           WHERE l0.idloc=l1.idloc AND NOT EXISTS(SELECT *
                                FROM rifiuto
                                WHERE l1.idloc=rifiuto.idloc));
```

8. Selezionare I gruppi di account con risorsa “risorsa” assegnata dal tecnico “tecnico”

```
SELECT AnnoIscrizione, IdCDL
FROM DISTRIBUZIONEPR
WHERE IdRis = IdRisorsa AND CF = 'cfTecnico'
```

9. Visualizzare gli intervalli orari in cui è permesso accedere al “Laboratorio”.

```
SELECT DISTINCT T.OraInizio, T.OraFine, L.IdLoc
FROM PERMESSO AS P, LOCALE AS L, TIPOLOGIA AS T
WHERE L.IdLoc = P.IdLoc AND T.IdT = P.IdT AND L.IdLoc =IdLab
```

10. Eliminare una prenotazione fatta da un utente;

```
DELETE FROM PRENOTAZIONE WHERE Data = 'Data' AND Ora ='Ora' AND
IdPosto = IdPosto AND IdLoc = IdLoc
```

11. Eliminare una Risorsa;

```
DELETE FROM RISORSA WHERE idRis=idRis;
```

12. Eliminare un utente;

```
DELETE FROM UTENTE WHERE CF='CFUTENTE';
```

13. Aggiornare un intervallo della tipologia di permesso;

```
UPDATE TIPOLOGIA SET OraInzio='OraiNIZIO',OraFine='OraFine'
WHERE TIPOLOGIA=IDTIPO;
```

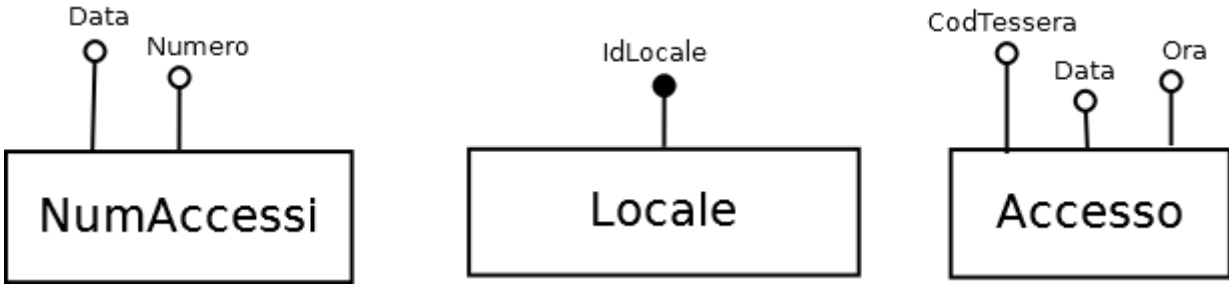
14. Aggiornare dati anagrafici utente;

```
UPDATE UTENTE SET
Nome='Nome',Cognome='Cognome',dataN='data',luogoN='luogo',residenza='resi
denza',CAP=CAP,Nazionalita='Naz',CodTessera=COD
WHERE CF='CFUTENTE';
```

15. Modificare posto di lavoro alle risorse;

```
UPDATE RISORSE SET idPosto=idposto,idloc=idloc WHERE  idRis=IDRIS;
```

10 - Dati derivati



CONCETTO	TIPO	VOLUME
Locale	E	50
Accesso	E	1000000
NumAccessi	E	40000

OPERAZIONE	TIPO	FREQUENZA
Nuovo accesso	I	1000/G
Visualizza accessi giornalieri	I	1/G

<u>Con dato derivato:</u>

CONCETTO	ACCESSO	TIPO
Accesso	1	S
Locale	1	L
NumAccessi	1	L
NumAccessi	1	S
NumAccessi	1	L

Operazione 1 : 2S +2L → 6*1000 = 6000/G
Operazione 2 : 1L → 1*1 = 1/G
Totale : 6001/G

<u>Senza dato derivato:</u>

CONCETTO	ACCESSO	TIPO
Accesso	1	S
Locale	1	L
Accesso	1000000/50 = 20000	L

Operazione 1: 1S → 2*1000 = 2000/G
Operazione 2: 20001L → 20001*1/G = 20001 /G
Totale : 22001/G

Conclusione :
Conviene tenere l'entità NumAccessi che contiene in nostro dato derivato poiché senza di esso sono previste 22001 operazioni al giorno, mentre ne sono previste solo 6001.

11 – Trigger

1) I laboratori hanno al massimo 100 posti di lavoro

```
CREATE FUNCTION controlloPosti() RETURNS trigger AS $$
      DECLARE
            CONT INTEGER;
      BEGIN
            SELECT COUNT(IdPosto) INTO CONT
            FROM PDLAVORO
            WHERE IdLoc = NEW.IdLoc;
            IF CONT > 100 THEN
                  RAISE EXCEPTION 'Laboratorio pieno';
            END IF;
            RETURN NEW;
      END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER CAPIENZALAB
AFTER INSERT ON PDLAVORO
FOR EACH ROW EXECUTE PROCEDURE controlloPosti();
```

Quando viene inserita una nuova risorsa il trigger controlla se nella tabella PDLAVORO vi sono più di 100 posti di lavoro per lo stesso laboratorio a cui sarebbe assegnata il nuovo posto di lavoro inserito. In caso positivo viene sollevata un'eccezione, altrimenti è consentito l'inserimento della nuova tupla.

2) Un posto di lavoro ha al massimo 20 risorse

```
CREATE FUNCTION controlloRisorse() RETURNS trigger AS $$
      DECLARE
            CONT INTEGER;
      BEGIN
            SELECT COUNT(IdRis) INTO CONT
            FROM RISORSA
            WHERE IdPosto = NEW.IdPosto;
            IF CONT > 20 THEN
                  RAISE EXCEPTION 'Impossibile assegnare ulteriore risorsa
al posto di lavoro';
            END IF;
            RETURN NEW;
      END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER NUMRISORSE
AFTER INSERT ON RISORSA
FOR EACH ROW EXECUTE PROCEDURE controlloRisorse();

CREATE TRIGGER NUMRISORSEONUPDATE
AFTER UPDATE ON RISORSA
FOR EACH ROW EXECUTE PROCEDURE controlloRisorse();
```

Quando viene inserita una nuova risorsa o aggiornata il trigger controlla se nella tabella RISORSA vi sono più di 20 risorse per lo stesso posto di lavoro a cui sarebbe assegnata la nuova risorsa inserita. In caso positivo viene sollevata un'eccezione, altrimenti è consentito l'inserimento o la modifica della nuova tupla.

3) La prenotazione di un posto di lavoro è impossibile per i non docenti e i docenti non titolari di un corso

```
CREATE FUNCTION controlloPrenotazioni() RETURNS trigger AS $$
      DECLARE
            APP   CHAR(16);
            APP2 CHAR(16) ;
            APP3 BOOLEAN;
      BEGIN
            SELECT CF INTO APP
            FROM NONDOCENTE
            WHERE CF = NEW.CF;

            IF (APP = NEW.CF) THEN
                  RAISE EXCEPTION 'Non è possibile prenotare';
            ELSE
                  SELECT CF, TITOLARE INTO APP2 ,  APP3
                  FROM DOCENTE
                  WHERE CF = NEW.CF;

                  IF (APP2 = NEW.CF AND APP3 = false ) THEN
                        RAISE EXCEPTION 'Non è possibile prenotare';
                  END IF;
            END IF;
            RETURN NEW;
      END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER CONTROLLOPRENOTAZIONE
AFTER INSERT ON PRENOTAZIONE
FOR EACH ROW EXECUTE PROCEDURE controlloPrenotazioni();
```

Quando viene inserita un nuova tupla nella tabella PRENOTAZIONE, il trigger controlla che essa sia accettabile, ovvero che non sia stata compiuta da un “non docente” o da un docente non titolare di un corso.
Inizialmente si salva in una variabile d'appoggio APP il CF del non docente con lo stesso CF della nuova tupla; se non è trovato APP avrà un valore nullo per cui la condizione successiva non sarà soddisfatta. Se invece APP ottenesse il valore del CF significherebbe che tale CF appartiene ad un non docente, per cui verrebbe sollevata un'eccezione.
Abbiamo usato tale soluzione per verificare l'esistenza di una certa tupla in una data tabella.
Se il primo controllo non solleva eccezione ve ne è un secondo. Con la stessa tecnica del primo si controlla se il CF nella tupla inserita appartiene alla tabella DOCENTE, in caso positivo si controlla se il flag TITOLARE è settato a falso : in tal caso viene nuovamente sollevata un'eccezione.
Se nessuna eccezione è sollevata significa che la prenotazione è stata fatta da un docente titolare di un corso o da uno studente, per cui è accettabile l'inserimento nella tabella.

4) E' possibile accedere ad un locale solo se si è in possesso di un permesso che lo permetta.

```
CREATE FUNCTION controlloACC() RETURNS trigger AS $$
      DECLARE
            APP CHAR(16);
            APP1 CHAR(16);
            APP2 INTEGER;
            OI  TIME;
            OF TIME;
            LOC INTEGER;
      BEGIN
            SELECT CF INTO APP1
            FROM UTENTE
            WHERE CodTessera = NEW.CodTessera;

            SELECT CF, IdT INTO  APP, APP2
            FROM PERMESSO
            WHERE CF  = APP1 AND IdLoc = NEW.IdLoc;

            SELECT OraInizio, OraFine INTO OI,OF
            FROM TIPOLOGIA
            WHERE IdT = APP2;

            IF(APP <> APP1 OR APP IS NULL OR NEW.Ora > OF OR NEW.Ora < OI)
THEN
                  INSERT INTO RIFIUTO VALUES
(NEW.Data,NEW.Ora,NEW.CodTessera, NEW.IdLoc);
                  RETURN NULL;
            ELSE

                  SELECT IdLoc INTO LOC
                  FROM NUMACCESSI
                  WHERE Data = NEW.Data;

                  IF(LOC = NEW.IdLoc) THEN
                        UPDATE NUMACCESSI
                        SET   Numero = Numero +1
                        WHERE IdLoc = NEW.IdLoc;
                  ELSE
                        INSERT INTO NUMACCESSI
VALUES(NEW.IdLoc,1,NEW.Data);
                  END IF;

                  RETURN NEW;
            END IF;

      END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER CONTROLLOACC
BEFORE INSERT ON ACCESSO
FOR EACH ROW EXECUTE PROCEDURE controlloACC();
```

Prima di inserire una nuova tupla in ACCESSO il trigger controlla che essa sia valida, in caso affermativo aggiorna il conteggio degli accessi giornalieri; in caso contrario il fallito accesso è inserito in RIFIUTO.
Inizialmente viene estratto dalla tabella UTENTE, in base al codice tessera con cui è stato compiuto il nuovo accesso, il cf e viene inserito in APP1. Dopodiché grazie ad esso vengono estratti il cf e il tipo di permesso per la coppia utente-locale dalla tabella PERMESSO e vengono posti in APP e APP2. Ciò viene fatto per verificare prima di tutto che tale utente abbia un permesso per tale locale. Infine vengono estratti dalla TIPOLOGIA di permessi l'orario di inizio e fine permesso, poi salvati nelle variabili di supporto OI e OF.
Nel controllo si verifica se

- APP sia diversa da APP1 o APP è nullo: ciò si verifica solo quando non vi è una tupla per quella coppia utente-locale : l'utente non ha nemmeno il permesso di entrarvi;
- L'ora di accesso sia inferiore all'OI;
- L'ora di accesso sia maggiore di OF.

Se solo una di esse non è valida l'accesso viene negato ed inserito in RIFIUTO, il trigger ritorna NULL e conclude la transazione (*"It can return NULL to skip the operation for the current row. This instructs the executor to not perform the row-level operation that invoked the trigger (the insertion, modification, or deletion of a particular table row)"*- www.postgresql.org/docs/9.2/static/trigger-definition.html).
Se invece tutte e tre le condizioni sono rispettate, si estrae il locale da NUMACCESSI per cui la data è la stessa dell'accesso inserito e si inserisce in LOC. Il controllo seguente verifica se LOC è uguale all'id del locale in cui è compiuto l'accesso valido. Ciò è vero solo se esiste già una tupla per quel locale: in tal caso viene solo modificata e il numero di accessi giornalieri viene incrementato di uno. In caso contrario nella tabella NUMACCESSI viene inserita una nuova tupla corrispondente a tale locale e il numero di accessi giornalieri viene posto ad 1.

5) La distribuzione di risorse ad un gruppo di account può essere eseguita solo da un Sistem Administrator

```
CREATE FUNCTION controlloSA() RETURNS trigger AS $$
      DECLARE
            APP CHAR(50);
      BEGIN
            SELECT RUOLO INTO APP
            FROM TECNICO
            WHERE CF  = NEW.CF;
            IF(APP <> 'System Administrator') THEN
                  RAISE EXCEPTION 'Tecnico non autorizzato';
            END IF;
            RETURN NEW;
      END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER CONTROLLOSA
AFTER INSERT ON DISTRIBUZIONEPR
FOR EACH ROW EXECUTE PROCEDURE controlloSA();
```

Quando si inserisce una nuova tupla in DISTRIBUZIONEPR tale trigger controlla che il tecnico coinvolto sia un System Administrator.
Esso estrae il ruolo del tecnico in questione inserendolo nella variabile di supporto APP. Dopodichè vi è un controllo : se il ruolo non è System Administrator allora viene sollevata un'eccezione. In caso contrario la nuova tupla è accettata.