Studente: Lorenzo Calisti

Matricola: 271250

Corso di Basi di Dati

Progetto per la sessione autunnale 2016/2017

Docente: Maurizio Maffi

Specifica Dei Requisiti

SPECIFICA DEI REQUISITI

Lo scopo principale della piattaforma è consentire agli utenti registrati di mantenere una lista di persone conosciute e ritenute affidabili in ambito lavorativo. Le persone nella lista sono definite "connessioni": esse sono in effetti le connessioni di un nodo (l'utente) all'interno della rete sociale. L'utente può incrementare il numero delle sue connessioni invitando chi di suo gradimento.

La rete di contatti a disposizione dell'utente è costituita da tutte le connessioni dell'utente, tutte le connessioni delle sue connessioni ("connessioni di secondo grado") e da tutte le connessioni delle connessioni di secondo grado ("connessioni di terzo grado").

Attraverso la piattaforma è possibile effettuare diverse attività:

1. essere presentati a qualcuno che si desidera conoscere attraverso un contatto mutuo e affidabile.
2. trovare offerte di lavoro, persone, opportunità di business con il supporto di qualcuno presente all'interno della propria lista di contatti o del proprio network.
3. i datori di lavoro (utenti premium, a pagamento) possono pubblicare offerte e ricercare potenziali candidati.
4. le persone in cerca di lavoro (utenti semplici, account free) possono consultare tali offerte nella sezione Offerte di lavoro.
5. gli utenti possono presentare la loro professionalità evidenziando nel proprio profilo personale (curriculum) i propri skill professionali e le competenze lavorative, oltre alle classiche informazioni presenti in un curriculum (dati anagrafici, studi, posizioni lavorative ricoperte etc.)
6. i gruppi sono formati da utenti che hanno qualcosa in comune, come un particolare percorso di carriera lavorativa, interessi di business simili, una specifica provenienza geografica o altro.
7. le segnalazioni sono veri e propri attestati di stima di chi ha lavorato con te – collega, datore di lavoro o cliente – e che riconosce il tuo talento, la tua preparazione, la tua professionalità.

Anche l'area del profilo dedicata ai riconoscimenti e ai premi ottenuti è importante per valorizzare la propria immagine di professionista agli occhi degli altri iscritti sul social network e delle aziende o dei brand che vi hanno stabilito una presenza.

Analisi Dei Requisiti

ANALISI DEI REQUISITI

La piattaforma deve garantire a chiunque la possibilità di accedere con le proprie credenziali ognuna delle quali definisce un univoco utente con pieno accesso ai servizi offerti.

Ogni utente possiede un suo profilo personale al cui interno sono riportate diverse informazioni personali, proprio come in un curriculum, quali: skill (capacità esterne al lavoro), competenze lavorative, posizioni lavorative ricoperte in passato, titoli di studio ottenuti, oltre ai dati anagrafici: sesso, data di nascita, luogo di nascita, luogo di residenza.

L'utente inoltre può essere di due tipi differenti: utente gratuito, ha accesso a tutti i servizi del sito, utente a pagamento: è come un utente gratuito solo che ha la possibilità di inserire gli annunci di lavoro. Una conseguenza di questa divisione degli utenti sono le informazioni sulla carta di credito necessarie solo all'utente premium per il pagamento. Nel profilo dell'utente vi è una sezione dedicata alle valutazioni degli altri utenti; ogni utente il quale ha collaborato con lui può inserire una valutazione professionale in modo da creare una bacheca di voti che identificano immediatamente le opinioni di colleghi e collaboratori a livello professionale.

Ogni utente ha un insieme di contatti da lui conosciuti; ognuno è detto connessione, dunque un'utente ha una lista di connessioni che può incrementare aggiungendo utenti non ancora presenti inviando a loro una richiesta di connessione che possono accettare o declinare.

Ogni utente può creare e/o fare parte di uno o più gruppi i quali non sono altro che una rete di utenti con delle caratteristiche in comune.

Ogni tipo di utente può consultare gli annunci inseriti all'interno della sua rete di connessioni (di primo, secondo e terzo grado), inoltre potrà vedere anche gli annunci inseriti da utenti facenti parte degli stessi gruppi cui lui appartiene.

Gli annunci possono essere inseriti solamente da utenti di tipo premium e consistono in un titolo, e una descrizione del tipo di lavoro offerto; tutti coloro che sono interessati all'annuncio possono dichiararlo e verranno inseriti in una lista di utenti interessati.

Glossario dei termini:

Termine	Descrizione	Sinonimo/Omonimo
Piattaforma	il prodotto finale che si vuole realizzare	applicazione
Utente	una qualsiasi persona che utilizza il servizio	account
Connessione	quando due utenti sono connessi tra di loro sono uno la connessione dell'altro	connessione 1° grado
Connessione 2° grado	una connessione di una connessione ovvero un utente connesso con qualcuno connesso a noi	-
Connessione 3°	una connessione di una connessione di 2° grado	-
Utente gratuito	l'utente che può accedere liberamente ai servizi, ma non può inserire annunci di lavoro	utente free/ utente normale
Utente premium	l'utente che può inserire annunci di lavoro	utente pro/ utente a pagamento/ datore di lavoro
Curriculum	contiene le informazioni personali e sulla carriera lavorativa dell'utente	-
Skill	sono delle capacità di un utente non strettamente collegate all'ambito lavorativo	-

Competenze lavorative	sono delle capacità di un utente apprese durante i lavori passati	-
Posizioni lavorative passate	i ruoli impiegati da un utente durante le esperienze lavorative passate	-
Titoli di studio	l'insieme di tutti i titoli di studio di un utente	-
Annuncio di lavoro	un annuncio inserito da un utente premium che cerca nuovi dipendenti	-
Gruppo	un insieme di utenti con caratteristiche in comune	-
Valutazione	un giudizio di un utente su un suo collaboratore	segnalazioni/ attestati di stima

L’applicazione deve essere in grado di eseguire le seguenti operazioni:

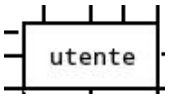
Interrogazioni	**Aggiornamenti**
Ottenere tutti gli annunci inseriti dalle connessioni	Aggiungere un nuovo utente
Ottenere tutti gli annunci inseriti dai gruppi	Aggiungere un nuovo annuncio
Verificare se un utente può inserire un annuncio	Rimuovere una competenza lavorativa di un utente
Verificare se due utenti sono già connessi tra di loro	Aggiornare i dati anagrafici di un utente
Ottenere tutte le informazioni su lavori passati di un utente	Aggiornare i dati account di un utente
Ottenere nome e id di tutte le connessioni di un utente	Aggiungere una valutazione
	Rimuovere una richiesta di amicizia

Progettazione Concettuale

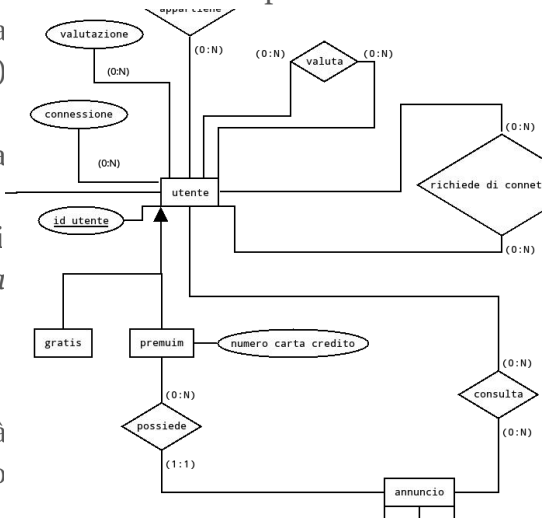
PROGETTAZIONE CONCETTALE

Per realizzare un buon diagramma E-R occorre partire dall'analisi dei requisiti e trovare tutte le entità, relazioni e attributi presenti.

L'utente, poiché descrive l'elemento chiave di tutta la piattaforma è un'entità la più importante e per questo è al centro dello schema e tutto il resto è in qualche modo collegato a lui.

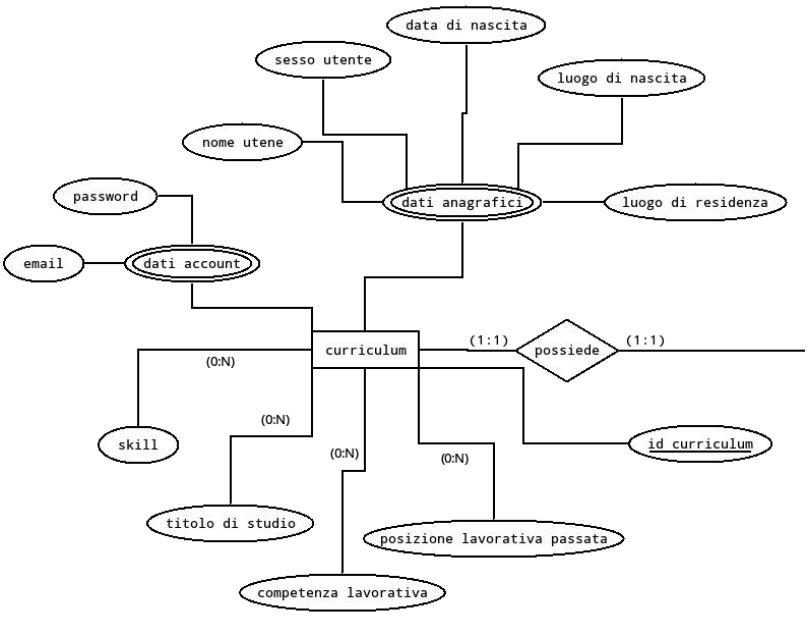


L'entità utente possiede un attributo *id utente,* il quale è un identificatore primario con lo scopo di identificare l'utente, due attributi a cardinalità multipla (0:N), uno (*connessione*) contenente le connessioni e l'altro (*valutazione*) le valutazioni degli utenti. L'utente possiede una generalizzazione stretta verso le entità *gratis* e *premium* le quali descrivono i due tipi di utente possibili l'entità premium gode di un attributo *numero carta credito* e di una connessione *possiede* verso l'entità annuncio.

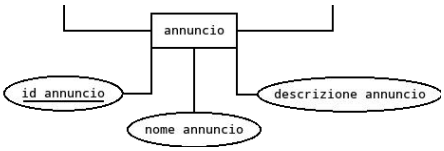


L'utente ha inoltre una relazione *consulta* verso l'entità annuncio e due relazioni molti a molti verso se stesso *valuta* e *richiede di connettersi.*

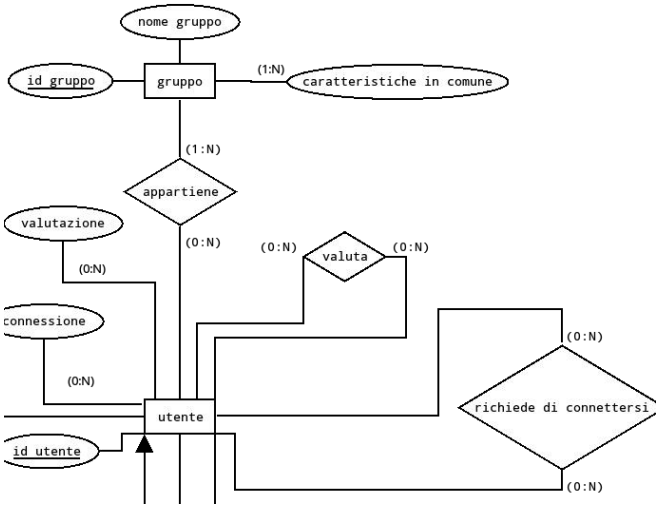
Come detto nell'analisi dei requisiti un utente possiede un unico curriculum ed è quindi legato ad esso da una relazione *possiede* uno a uno. L'entità curriculum è la più complessa poiché è composta da svariati attributi normali, multi-valore e a multi-cardinalità. Il curriculum contiene l'attributo *id curriculum* che è l'identificatore primario e ha lo scopo di individuare l'entità, *dati anagrafici* è un attributo multi-valore con al suo interno cinque altri attributi: *nome utente, sesso utente, data di nascita, luogo di nascita, luogo di residenza;* come dati anagrafici anche *dati account* è un attributo multi-valore con due attributi normali: *email, password*. L'entità curriculum ha poi quattro attributi a cardinalità multipla (0:N): *skill, titolo di studio competenza lavorativa, posizione lavorativa passata*.



L'entità *annuncio* esprime l'annuncio di lavoro inserito da un utente premium ed è composto da tre attributi normali: *id annuncio* è l'identificatore primario, *nome annuncio, descrizione annuncio.*

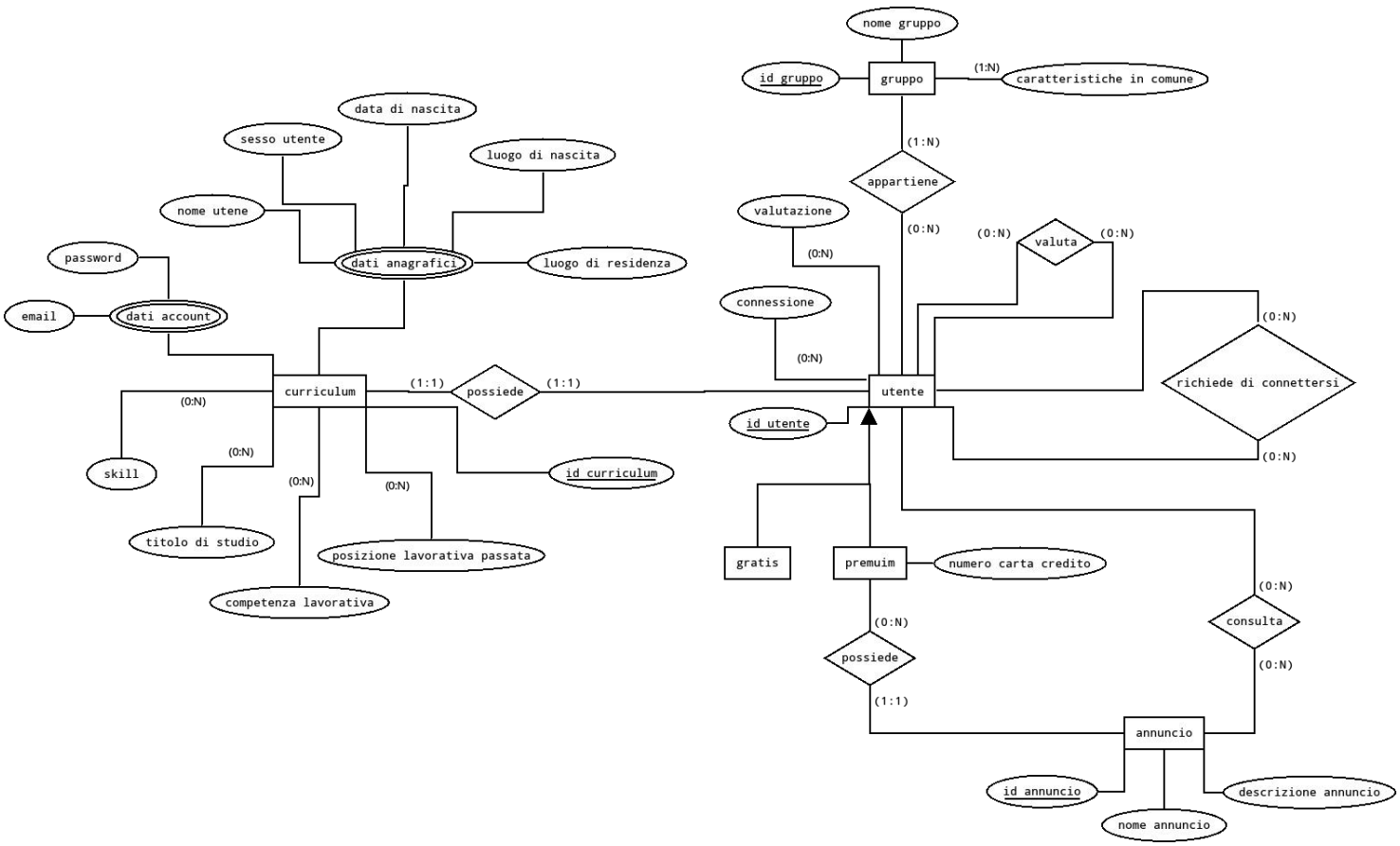


L'entità gruppo descrive il gruppo e possiede tre attributi, *id gruppo,* l'identificatore primario, *nome gruppo* e *caratteristica comune* la quale è un attributo a cardinalità multipla (1:N), poiché ogni gruppo come visto nell'analisi dei requisiti deve avere almeno una caratteristica in comune. L'utente è collegato all'entità gruppo tramite una relazione *appartiene* zero a molti (0:N) dal lato dell'utente e da una uno a molti (1:N) dal lato del gruppo



L'entità utente possiede infine una relazione *richiede di connettersi* la quale è una molti a molti (0:N)(0:N) ed una relazione *valuta*, sempre molti a molti, con il fine di inserire nuovi elementi negli attributi *connessione* e *valutazione.*

Di seguito è mostrato il diagramma E-R completo



Progettazione Logica

PROGETTAZIONE LOGICA

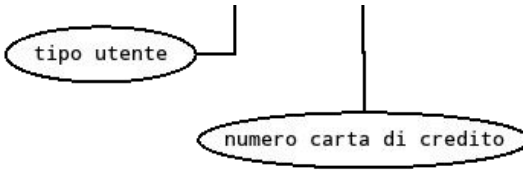
La progettazione logica si divide in due parti:

1. Ristrutturazione dello schema E-R
2. Traduzione dello schema nel modello logico

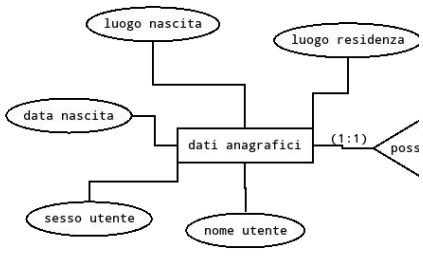
RISTRUTTURAZIONE DELLO SCHEMA E-R

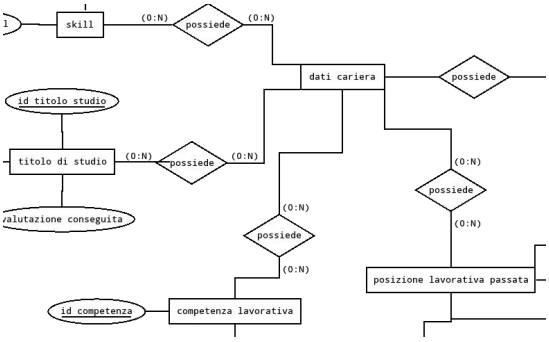
Per ristrutturare uno schema E-R occorre innanzitutto analizzare le ridondanze presenti nel vecchio schema per rimuovere entità o attributi con lo stesso significato; fortunatamente analizzando lo schema non sono presenti ridondanze di alcun tipo.

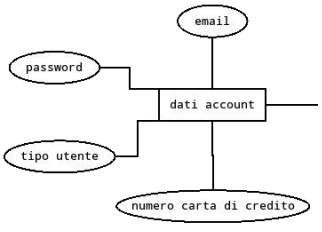
Una volta rimosse le ridondanze occorre eliminare le generalizzazioni; in questo caso nello schema è presente solo una generalizzazione tra i due tipi di utente, per questo motivo è opportuno applicare un accorpamento delle entità figlie nell'entità padre creando un attributo con il tipo d'utente ed uno per contenere il numero della carta di credito (nel caso in cui l'utente sia gratuito questo valore risulterà nullo, ma è un piccolo prezzo a confronto dello spreco che avverrebbe utilizzando ad esempio un accorpamento del genitore nelle figlie).



Una volta eliminate tutte le generalizzazioni occorre partizionare o accorpare le associazioni; in questa parte lo schema subirà molti cambiamenti. L'entità curriculum è stata suddivisa in tre sotto entità: *dati anagrafici, dati account, dati carriera*. L'entità dati anagrafici rappresenta quello che prima era l'omonimo attributo composto e contiene i suoi precedenti attributi. L'entità dati account sostituisce l'attributo composto dati account e al suo interno racchiude l'attributo email, password e accorpa il tipo d'utente e il numero della carta di credito. L'entità dati carriera invece è utilizzata come rapido accesso verso altre entità, è infatti legata tramite relazioni molti a molti a quattro entità: *skill, titolo di studio, competenza lavorativa, posizione lavorativa passata*. Gli attributi a cardinalità multipla connessione e valutazione di utente non potendo essere implementati in questo modo sono stati sostituiti da due relazioni *possiede* verso due entità *connessione* e *valutazione;* l'attributo a cardinalità multipla *caratteristiche in comune* è stato invece sostituito da un'entità omonima connessa al gruppo tramite una relazione molti a molti *possiede*. La relazione *richiede di connettersi* è stata trasformata in un entità *richiesta di connessione* la quale ha un attributo per l'utente



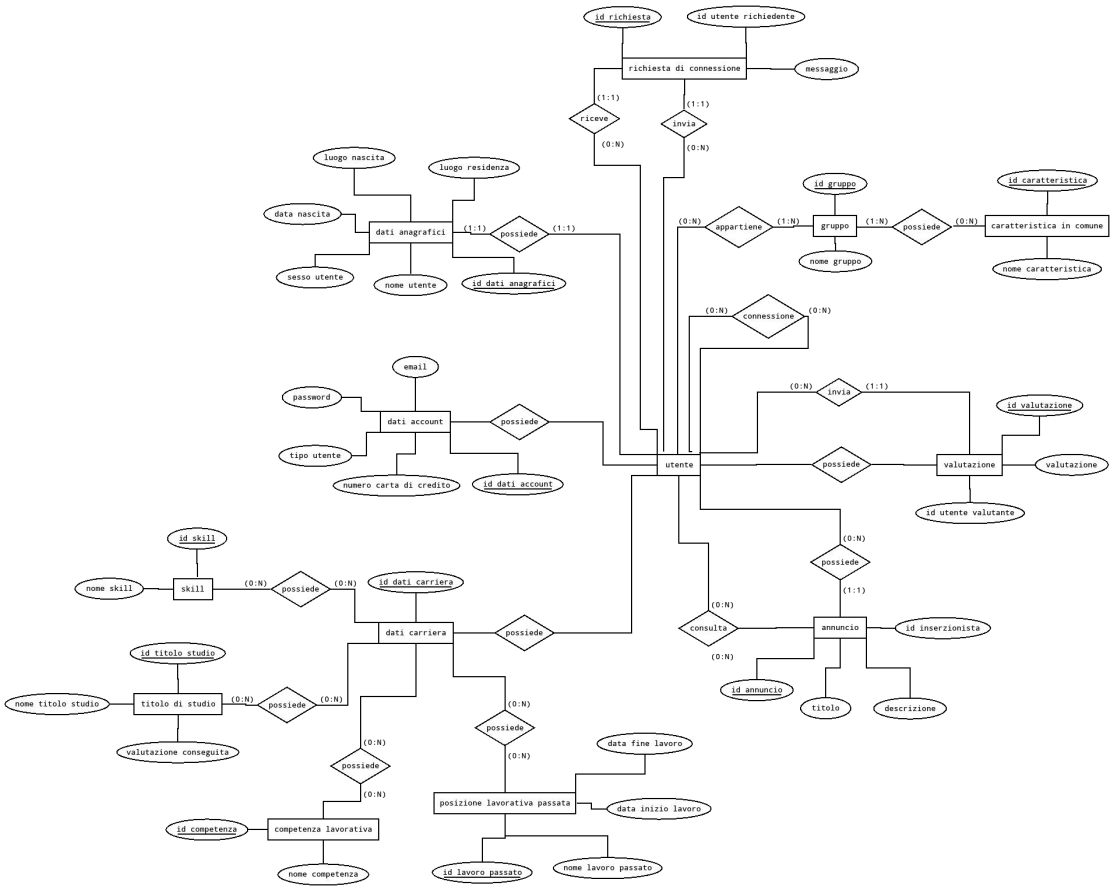




che effettua la richiesta e uno per il messaggio di descrizione della richiesta; la richiesta è collegata all'utente tramite due relazioni uno molti *riceve* e *invia*.

Una volta accorpate o partizionate le entità e gli attributi non resta altro che ridefinire gli identificatori principali: l'utente mantiene come identificatore primario *id utente,* all'entità dati anagrafici è dato un identificatore *id dati anagrafici*, all'entità dati carriera *id dati carriera* e all'entità dati account *id dati account*; le entità skill, competenza lavorativa, titolo di studio e posizione lavorativa passata è assegnato identificatore *id skill, id competenza, id titolo studio, id lavoro passato*. L'entità annuncio contiene l'identificatore *id annuncio*, la valutazione *id valutazione* e la richiesta di connessione *id richiesta*. Il gruppo ha come identificatore *id gruppo* e l'entità caratteristica in comune *id caratteristica*.

Di seguito è mostrato il nuovo diagramma E-R dopo la ristrutturazione.



TRADUZIONE DELLO SCHEMA NEL MODELLO LOGICO

La seconda fase della progettazione logica consiste nel tradurre il diagramma E-R appena ristrutturato in uno schema logico equivalente.

Partendo dall'entità principale convertiamo gli identificatori principali e gli identificatori esterni in modo da ottenere una lista con per ogni entità tutti i suoi attributi e le sue chiavi; l'entità utente è convertita in:

utente(idUtente, idDatiAnagrafici, idDatiAccount, idDatiCarriera)

L'entità dati anagrafici è convertita in:

datiAnagrafici(idDatiAnagrafici, nomeUtente, sesso, dataNascita, luogoNascita, luogoResidenza)

L'entità dati account diventa:

datiAccount(idDatiAccount, email, password, tipoUtente, numCartaCredito)

L'entità dati carriera:

datiCarriera(idDatiCarriera)

Le entità che rappresentano la carriera diventano:

skill(idSkill, nomeSkill)

lavoroPassato(idLavoroPassato, nomeLavoroPassato, dataInizio, dataFine)

titoloStudio(idTitoloStudio, valutazioneConseguita)

competenza(idCompetenza, nomeCompetenza)

L'annuncio di lavoro diventa:

annuncio(idAnnuncio, nomeAnnuncio, descrizioneAnnuncio, idUtente)

Il gruppo:

gruppo(idGruppo, nomeGruppo)

competenzaComune(idCompetenza, nomeCompetenza)

La valutazione è tradotta in:

valutazione(idValutazione, valutazione, idUtenteValutante)

La richiesta di connessione è tradotta in:

richiestaConnessione(idRichiesta, idUtenteRichiedente, messaggioRichiesta)

Ricapitolando gli elementi del modello E-R sono stati tradotti in:

utente(idUtente, idDatiAnagrafici, idDatiAccount, idDatiCarriera)

datiAnagrafici(idDatiAnagrafici, nomeUtente, sesso, dataNascita, luogoNascita, luogoResidenza)

datiAccount(idDatiAccount, email, password, tipoUtente, numCartaCredito)

datiCarriera(idDatiCarriera)

skill(idSkill, nomeSkill)

lavoroPassato(idLavoroPassato, nomeLavoroPassato, dataInizio, dataFine)

titoloStudio(idTitoloStudio, valutazioneConseguita)

competenza(idCompetenza, nomeCompetenza)

annuncio(idAnnuncio, nomeAnnuncio, descrizioneAnnuncio, idUtente)

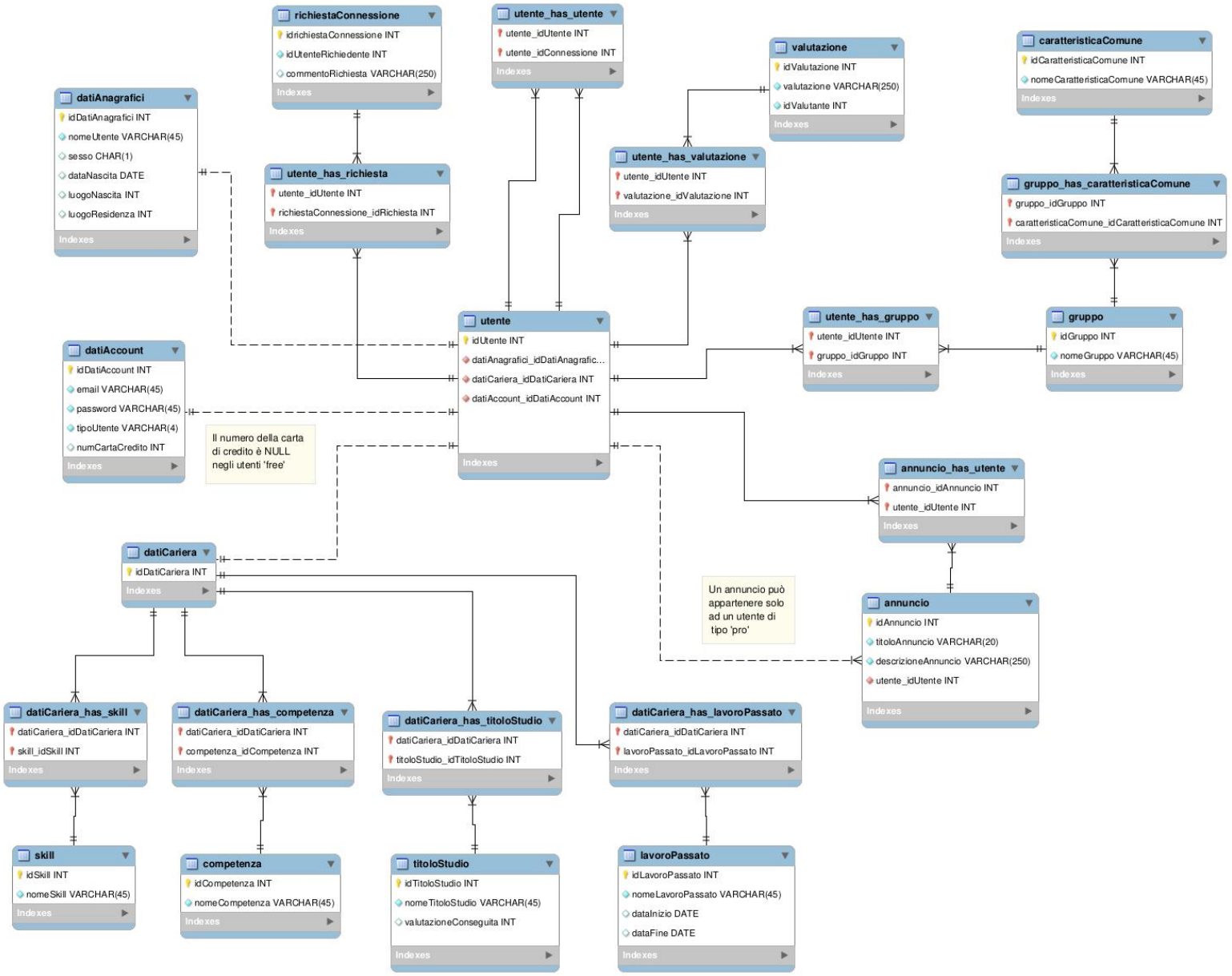
gruppo(idGruppo, nomeGruppo)

competenzaComune(idCompetenza, nomeCompetenza)

valutazione(idValutazione, valutazione, idUtenteValutante)

richiestaConnessione(idRichiesta, idUtenteRichiedente, messaggioRichiesta)

Una volta convertiti tutti gli elementi del diagramma siamo pronti a produrre il diagramma logico


richiestaConnessione
idrichiestaConnessione INT
idUtenteRichiedente INT
commentoRichiesta VARCHAR(250)
Indexes
utente_has_utente
utente_idUtente INT
utente_idConnessione INT
Indexes
valutazione
idValutazione INT
valutazione VARCHAR(250)
idValutante INT
Indexes
caratteristicaComune
idCaratteristicaComune INT
nomeCaratteristicaComune VARCHAR(45)
Indexes
datiAnagrafici
idDatiAnagrafici INT
nomeUtente VARCHAR(45)
sesso CHAR(1)
dataNascita DATE
luogoNascita INT
luogoResidenza INT
Indexes
utente_has_richiesta
utente_idUtente INT
richiestaConnessione_idRichiesta INT
Indexes
utente_has_valutazione
utente_idUtente INT
valutazione_idValutazione INT
Indexes
gruppo_has_caratteristicaComune
gruppo_idGruppo INT
caratteristicaComune_idCaratteristicaComune INT
Indexes
utente
idUtente INT
datiAnagrafici_idDatiAnagrafic...
datiCariera_idDatiCariera INT
datiAccount_idDatiAccount INT
Indexes
utente_has_gruppo
utente_idUtente INT
gruppo_idGruppo INT
Indexes
gruppo
idGruppo INT
nomeGruppo VARCHAR(45)
Indexes
datiAccount
idDatiAccount INT
email VARCHAR(45)
password VARCHAR(45)
tipoUtente VARCHAR(4)
numCartaCredito INT
Indexes
Il numero della carta di credito è NULL negli utenti 'free'
annuncio_has_utente
annuncio_idAnnuncio INT
utente_idUtente INT
Indexes
datiCariera
idDatiCariera INT
Indexes
Un annuncio può appartenere solo ad un utente di tipo 'pro'
annuncio
idAnnuncio INT
titoloAnnuncio VARCHAR(20)
descrizioneAnnuncio VARCHAR(250)
utente_idUtente INT
Indexes
datiCariera_has_skill
datiCariera_idDatiCariera INT
skill_idSkill INT
Indexes
datiCariera_has_competenza
datiCariera_idDatiCariera INT
competenza_idCompetenza INT
Indexes
datiCariera_has_titoloStudio
datiCariera_idDatiCariera INT
titoloStudio_idTitoloStudio INT
Indexes
datiCariera_has_lavoroPassato
datiCariera_idDatiCariera INT
lavoroPassato_idLavoroPassato INT
Indexes
skill
idSkill INT
nomeSkill VARCHAR(45)
Indexes
competenza
idCompetenza INT
nomeCompetenza VARCHAR(45)
Indexes
titoloStudio
idTitoloStudio INT
nomeTitoloStudio VARCHAR(45)
valutazioneConseguita INT
Indexes
lavoroPassato
idLavoroPassato INT
nomeLavoroPassato VARCHAR(45)
dataInizio DATE
dataFine DATE
Indexes

Normalizzazione

NORMALIZZAZIONE

In questa fase della relazione verificheremo se il modello appena realizzato rispetta le varie forme normali.

- **Verifica della seconda forma normale 2NF**

"Una relazione è in seconda forma normale (2NF) se su di essa non sono definite dipendenze parziali, cioè dipendenze fra un sottoinsieme proprio della chiave e altri attributi."

Ovvero una relazione per essere in seconda forma normale non deve avere nessun capo ripetuto, nessun attributo composto e nessun attributo che non appartiene all'intera chiave.

Prendiamo come esempio la tabella *datiAccount*:

idDatiAccount	**email**	**password**	**tipoUtente**	**numCartaCredito**
1	giorgio.f@libero.it	Giorgio25	free	-
2	l.calisti@gmail.it	lorenzoc	pro	0001564521897854
3	luca-m@alice.it	lucaemme	pro	0132456789520001

Possiamo notare che nessuna colonna è ripetuta più volte, non vi sono attributi composti, ne elementi che non dipendono dalla chiave *idDatiAccount*. Siamo in grado quindi di dire che questa tabella risulta rispettare la seconda forma normale.

Applicando la stessa metodologia a tutte le tabelle del database risultano rispettare la 2NF.

- **Verifica della terza forma normale 3NF**

"*Una relazione si dice in terza forma normale (3NF) se su di essa non sono definite dipendenze transitive, ciò dipendenze della forma K->A, dove K è la chiave ed esiste in un altro insieme di attributi X, non chiave, con le dipendente K->X->A."*

Ovvero una relazione è in terza forma normale se rispetta le precedenti e tutti gli attributi che non fanno parte della chiave dipendono solo da essa.

Verifichiamo la 3NF sulle tabelle *utente*, *utente_has_gruppo, gruppo*

Tabella utente

idUtente	**idDatiAnagrafici**	**idDatiAccount**	**idDatiCarriera**
1	1	1	1
2	2	2	2

Tabella utente_has_gruppo

idUtente	**idGruppo**
1	1
1	2
2	1

Tabella gruppo

idGruppo	**nomeGruppo**
1	Gruppo uno
2	Gruppo due

Come possiamo notare la tabella *utente_has_gruppo* ha come idUtente un valore che fa riferimento alla tabella *utente* e come idGruppo un valore che si riferisce alla tabella *gruppo*, dunque entrambi gli attributi non sono chiavi e sono legati ad una sola chiave; nella tabella gruppo l'attributo nomeGruppo dipende solo dalla chiave idGruppo e nella tabella utente gli attributi idDatiAnagrafici, idDatiAccount e idDatiCarriera dipendono da altre omonime tabelle. Per questo motivo possiamo dire che queste tabelle rispettano la terza forma normale.

Applicando questo metodo di analisi alle restanti tabelle risultano essere tutte in 3NF

- **Verifica della forma normale di Boyce-Codd**

“Una relazione è in forma normale di Boyce-Codd (BCNF) se rispetta le precedenti forme normali e, per ogni dipendenza funzionale non banale X->Y X(con Y non in X), X è una super-chiave per la relazione.

Dato un insieme di relazioni, non è possibile garantire sempre il raggiungimento della BCNF; in particolare il mancato raggiungimento di questo obiettivo è indice che la base dati è affetta da un'anomalia di cancellazione (ossia è possibile perdere dati a seguito di un'operazione di cancellazione).”

Verifichiamo la BCFN sulle tabelle *datiCarriera, carriera_has_titoloStudio e titoloStudio*

Tabella datiCarriera

idDatiCarriera
1
2

Tabella carriera_has_titoloStudio

idDatiCarriera	**idTitoloStudio**
1	1
1	2
2	1

Tabella titoloStudio

idTitoloStudio	**nomeTitoloStudio**	**valutazioneConseguita**
1	Laurea in Economia	95
2	Liceo Scientifico	74

Come possiamo notare la colonna idDatiCarriera nella tabella *carriera_has_titoloStudio* identifica la chiave primaria della tabella datiCarriera la quale è una super-chiave essendo l'unico attributo presente; la colonna idTitoloStudio della tabella omonima è anch'essa una super-chiave poiché tutti i suoi attributi sono univocamente identificati da essa. Le tre tabelle dunque risultano essere in forma normale di Boyce-Codd.

Applicando la stessa metodologia di verifica alle restanti tabelle constatiamo che tutte risultano rispettare la BCNF.

Database MySql

DATABASE MYSQL

L'implementazione del diagramma logico in un database MySQL è estremamente semplice poiché tutte le entità diventano tabelle separate e i loro attributi diventano colonne in queste tabelle; l'unica cosa degna di nota in questa fase è l'implementazione delle relazioni molti a molti, per esse infatti occorre deviare dallo schema appena creato introducendo una nuova tabella intermedia contenente le chiavi esterne delle due entità in gioco nella relazione.

Le relazioni molti a molti tra datiCarriera e skill, lavoroPassato, competenza e titoloStudio subiscono l'introduzione di quattro tabelle intermedie carriera_has_skill, carriera_has_titoloStudio, carriera_has_competenza e carriera_has_lavoroPassato le quali al loro interno contengono una colonna idCarriera che identifica i datiCarriera e una colonna idSkill, idCompetenza, idLavoroPassato o idTitoloStudio che identifica l'altra tabella connessa.

Tramite lo stesso metodo implementiamo anche la relazione molti a molti tra utente e gruppo con una tabella utente_has_gruppo(idUtente, idGruppo), la relazione tra gruppo e le sue caratteristiche in comune gruppo_has_carComune(idGruppo, idCarComune), la relazione tra l'utente e le sue valutazioni utente_has_valutazione(idUtente, idValutazione), la relazione tra utente e le sue connessioni utente_has_utente(idUtente, idUtenteConnesso), la relazione tra utente e la richiesta di connessione utente_has_richiesta(idUtente, idRichiesta) e la relazione tra annuncio e utente interessato annuncio_has_utente(idAnnuncio, idUtente).

Una volta implementato il database avrà una struttura del tipo:

Implementato il database possiamo realizzare le operazioni di inserimento e aggiornamento dei dati descritte nelle sezioni precedenti:

Ottenere tutti gli annunci inseriti dalle connessioni

```
SELECT annuncio.*, datiAnagrafici.nomeUtente FROM (SELECT c1.idConnessione FROM utente
_has_utente AS c1 WHERE c1.idUtente=ID-UTENTE UNION SELECT u2.idConnessione FROMutente
_has_utente AS u1 INNER JOIN utente_has_utente AS u2 ON u1.idConnessione=u2.idUtente W
HERE u2.idConnessione<>u1.idUtente AND u1.idUtente=ID-UTENTE UNION SELECTu3.idConnessi
one FROM utente_has_utente AS u1 INNER JOIN utente_has_utente AS u2 ON u1.idConnession
e=u2.idUtente INNER JOIN utente_has_utente AS u3 ONu2.idConnessione=u3.idUtente WHERE
 u3.idConnessione<>u1.idUtente AND u1.idUtente=ID-UTENTE) AS conn INNER JOIN annuncio O
N conn.idConnessione=annuncio.idUtente INNER JOIN utenteON annuncio.idUtente=utente.id
Utente INNER JOIN datiAnagrafici ON utente.idDatiAnagrafici=datiAnagrafici.idDatiAnagr
afici
```

Ottenere tutti gli annunci inseriti dai gruppi

```
SELECT tab2.idGruppo, gruppo.nomeGruppo, annuncio.*, datiAnagrafici.nomeUtente FROM ut
ente_has_gruppo AS tab1 INNER JOIN utente_has_gruppo as tab2 ONtab1.idGruppo=tab2.idGr
uppo INNER JOIN annuncio ON tab2.idUtente=annuncio.idUtente INNER JOIN gruppo ON tab2.
idGruppo=gruppo.idGruppo INNER JOIN utente ONtab2.idUtente=utente.idUtente INNER JOIN
 datiAnagrafici ON utente.idDatiAnagrafici=datiAnagrafici.idDatiAnagrafici WHERE tab2.i
dUtente!=tab1.idUtente AND tab1.idUtente=ID-UTENTE;
```

Ottenere nome e id di tutte le connessioni di un utente

```
SELECT uu.idConnessione, datiAnagrafici.nomeUtente FROM utente_has_utente AS uu INNER
 JOIN utente on uu.idConnessione=utente.idUtente NATURAL JOIN datiAnagrafici WHEREuu.id
Utente=ID-UTENTE;
```

Verificare se un utente può inserire un annuncio

```
SELECT acc.tipoUtente FROM utente NATURAL JOIN datiAccount AS acc WHERE utente.idUtent
e=ID-UTENTE;
```

Verificare se due utenti sono già connessi tra di loro

```
SELECT (CASE WHEN EXISTS (SELECT * FROM utente_has_utente AS uu WHERE uu.idUtente=ID-U
TENTE AND uu.idConnessione=ID-AMICO) THEN 'Si' ELSE 'No' END) AS 'Sono già amici?';
```

Ottenere tutte le informazioni su lavori passati di un utente

```
SELECT lavoroPassato.* FROM utente NATURAL JOIN datiCarriera NATURAL JOIN carriera_has
_lavoroPassato NATURAL JOIN lavoroPassato WHERE utente.idUtente=ID-UTENTE;
```

Aggiornare i dati anagrafici di un utente

```
UPDATE datiAnagrafici AS anag SET anag.nomeUtente=NOME-UTENTE, anag.sesso=SESSO-UTENTE, anag.dataNascita=DATA-NASCITA, anag.luogoNascita=LUOGO-NASCITA, anag.luogoResidenza=LUOGO-RESIDENZA WHERE anag.idDatiAnagrafici=(SELECT u.idDatiAnagrafici FROM utente AS u WHERE u.idUtente=ID-UTENTE);
```

Aggiungere un nuovo annuncio

```
INSERT INTO annuncio (annuncio.titoloAnnuncio, annuncio.descrizioneAnnuncio, annuncio.idUtente) VALUES (TITOLO, DESCRIZIONE, ID-INSERZIONISTA);
```

Aggiungere un utente

```
START TRANSACTION
INSERT INTO datiAnagrafici (datiAnagrafici.nomeUtente, datiAnagrafici.sesso, datiAnagrafici.dataNascita, datiAnagrafici.luogoNascita, datiAnagrafici.luogoResidenza)VALUES ('NOME', 'M', '2017-09-01', 'ROMA', 'TORINO')
SET @anag = LAST_INSERT_ID()
INSERT INTO datiAccount (datiAccount.email, datiAccount.password, datiAccount.tipoUtente, datiAccount.numCartaCredito) VALUES ('E@M', 'PASS', 'PRO', '4567')
SET @acc = LAST_INSERT_ID()
INSERT INTO datiCarriera () VALUES ()
SET @carr = LAST_INSERT_ID()
INSERT INTO utente (utente.idDatiAnagrafici, utente.idDatiCarriera, utente.idDatiAccount) VALUES (@anag, @carr, @acc)
SELECT LAST_INSERT_ID() AS idUtente
COMMIT;
```

Rimuove una competenza lavorativa di un utente

```
DELETE FROM carriera_has_competenza WHERE idCompetenza=ID-COMPETENZA AND idDatiCarriera= (SELECT carr.idDatiCarriera FROM datiCarriera AS carr NATURAL JOIN utente WHEREutente.idUtente=ID-UTENTE);
```

Aggiornare i dati account di un utente

```
UPDATE datiAccount AS acc SET acc.email=EMAIL, acc.password=PASSWORD, acc.tipoUtente=TIPO-UTENTE, acc.numCartaCredito=NUM-CARTA WHERE acc.idDatiAccount = (SELECT u.idDatiAccount FROMutente AS u WHERE u.idUtente=ID-UTENTE);
```

Aggiungere una valutazione

```
START TRANSACTION
INSERT IGNORE INTO
valutazione (valutazione.valutazione, valutazione.idValutante) VALUES (VALUTAZIONE, ID-VALUTANTE);
```

```
SET @idVal = (SELECT valutazione.idValutazione FROM valutazione WHERE valutazione.valu
tazione=VALUTAZIONE AND valutazione.idValutante=ID-VALUTANTE)


INSERT IGNORE INTO utente_has_valutazione (utente_has_valutazione.idValutato,
utente_has_valutazione.idValutazione) VALUES (ID-UTENTE, @idVal);
COMMIT;
```

Rimuovere una richiesta di amicizia

```
START TRANSACTION
SET @idR = (SELECT tab.idRichiesta FROM (SELECT * FROM richiestaConnessione) AS tab NA
TURAL JOIN (SELECT * FROM utente_has_richiesta) AS ur WHERE tab.idUtenteRichiedente =
ID-CONNESSIONE AND ur.idUtente=ID-UTENTE);
DELETE richiestaConnessione.* FROM richiestaConnessione WHERE richiestaConnessione.idR
ichiesta = @idR;
DELETE utente_has_richiesta.* FROM utente_has_richiesta WHERE utente_has_richiesta.idR
ichiesta = @idR AND utente_has_richiesta.idUtente = ID-UTENTE;
COMMIT;
```

Applicazione Client

APPLICAZIONE CLIENT

Una volta progettata e realizzata la struttura del database si è scelto di realizzare un applicazione client in grado di gestire i dati. L'applicazione è stata realizzata in PHP(lato server) e HTML/CSS/JavaScript (lato browser).

Il sito consente il login degli utenti e la loro registrazione, l'utente una volta entrato può consultare tutti gli annunci delle sue connessioni o dei suoi gruppi, può inserire degli annunci, può seguire degli annunci in particolare, aggiungere nuovi utenti alle sue connessioni o creare o partecipare a nuovi gruppi; l'utente infine può visualizzare il profilo di altri utenti e inserire delle valutazioni o può visionare e modificare il suo profilo.

In PHP sono state utilizzate tutte le query descritte e implementate nelle precedenti sezioni.

Ottenere tutti gli annunci inseriti dalle connessioni

```
function getAnnunciConnessioni($idUtente){

    $link = connectToDatabase();

    $query_amici = 'SELECT annuncio.*, datiAnagrafici.nomeUtente FROM (SELECT
c1.idConnessione FROM utente_has_utente AS c1 WHERE c1.idUtente='.$idUtente.' UNION SELECT
u2.idConnessione FROM utente_has_utente AS u1 INNER JOIN utente_has_utente AS u2 ON
u1.idConnessione=u2.idUtente WHERE u2.idConnessione<>u1.idUtente AND
u1.idUtente='.$idUtente.' UNION SELECT u3.idConnessione FROM utente_has_utente AS u1 INNER
JOIN utente_has_utente AS u2 ON u1.idConnessione=u2.idUtente INNER JOIN utente_has_utente AS
u3 ON u2.idConnessione=u3.idUtente WHERE u3.idConnessione<>u1.idUtente AND
u1.idUtente='.$idUtente.') AS conn INNER JOIN annuncio ON conn.idConnessione=annuncio.idUtente
INNER JOIN utente ON annuncio.idUtente=utente.idUtente INNER JOIN datiAnagrafici ON
utente.idDatiAnagrafici=datiAnagrafici.idDatiAnagrafici';

    $res = mysqli_query($link, $query_amici);

    $annunci = array();

    while ($row = mysqli_fetch_assoc($res))

        $annunci[] = $row;

    disconnectFromDatabase($link);

    return $annunci;

}
```

Ottenere tutti gli annunci inseriti dai gruppi

```
function getAnnunciGruppi($idUtente){
```

```
	$link = connectToDatabase();

	$query_gruppi = 'SELECT tab2.idGruppo, gruppo.nomeGruppo, annuncio.*,
datiAnagrafici.nomeUtente FROM utente_has_gruppo AS tab1 INNER JOIN utente_has_gruppo as
tab2 ON tab1.idGruppo=tab2.idGruppo INNER JOIN annuncio ON tab2.idUtente=annuncio.idUtente
INNER JOIN gruppo ON tab2.idGruppo=gruppo.idGruppo INNER JOIN utente ON
tab2.idUtente=utente.idUtente INNER JOIN datiAnagrafici ON
utente.idDatiAnagrafici=datiAnagrafici.idDatiAnagrafici WHERE tab2.idUtente!=tab1.idUtente AND
tab1.idUtente='.$idUtente;

	$res = mysqli_query($link, $query_gruppi);

	$annunci_gruppi = array();

	while ($row = mysqli_fetch_assoc($res))

		$annunci_gruppi[] = $row;

	disconnectFromDatabase($link);

	return $annunci_gruppi;

}
```

Ottenere nome e id di tutte le connessioni di un utente

```
	 $res = mysqli_query($link, 'SELECT uu.idConnessione, datiAnagrafici.nomeUtente    FROM
utente_has_utente AS uu INNER JOIN utente on uu.idConnessione=utente.idUtente    NATURAL JOIN
datiAnagrafici WHERE uu.idUtente='.$idUtente);

	$connessioni = array();

	while ($row = mysqli_fetch_assoc($res))

		$connessioni[] = $row;
```

Verificare se un utente può inserire un annuncio

```
function canUserCreateAnnuncio($idUtente){

	$link = connectToDatabase();

	$res = mysqli_query($link, 'SELECT acc.tipoUtente FROM utente NATURAL JOIN datiAccount AS
acc WHERE utente.idUtente='.$idUtente);

	$tipoU = mysqli_fetch_assoc($res);

	disconnectFromDatabase($link);

	if ($tipoU['tipoUtente'] == 'pro')
```

```
        return true;
    else
        return false;
}
```

Verificare se due utenti sono già connessi tra di loro

```
function areFriends($idUtente, $idAmico){
    $link = connectToDatabase();
    mysqli_query($link, 'SELECT * FROM utente_has_utente AS uu WHERE uu.idUtente='.$idUtente.' AND uu.idConnessione='.$idAmico);
    $aff_rows = mysqli_affected_rows($link);
    disconnectFromDatabase($link);
    if ($aff_rows > 0)
        return true;
    else
        return false;
}
```

Ottenere tutte le informazioni su lavori passati di un utente

```
function getLavoriPassati(){
    $link = connectToDatabase();
    $res = mysqli_query($link, 'SELECT lavoroPassato.* FROM utente NATURAL JOIN datiCarriera NATURAL JOIN carriera_has_lavoroPassato NATURAL JOIN lavoroPassato WHERE utente.idUtente='.getUserID());
    $lavoriPassati = array();
    while ($row = mysqli_fetch_assoc($res))
        $lavoriPassati[] = $row;
    disconnectFromDatabase($link);
    return $lavoriPassati;
}
```

Aggiornare i dati anagrafici di un utente

```
function updateDatiAnagrafici($valori){

    $link = connectToDatabase();

    $stmt = mysqli_prepare($link, 'UPDATE datiAnagrafici AS anag SET anag.nomeUtente=?, anag.sesso=?,anag.dataNascita=?, anag.luogoNascita=?, anag.luogoResidenza=? WHERE anag.idDatiAnagrafici=(SELECT u.idDatiAnagrafici FROM utente AS u WHERE u.idUtente=?)');

    mysqli_stmt_bind_param($stmt, 'sssssi', $valori['nomeUtente'], $valori['sesso'], $valori['dataN'], $valori['luogoN'], $valori['luogoR'], getUserID());

    mysqli_stmt_execute($stmt);

    mysqli_stmt_close($stmt);

    disconnectFromDatabase($link);

}
```

Aggiungere un nuovo annuncio

```
function creaAnnuncio($titolo, $descrizione, $idCreatore){

    $link = connectToDatabase();

    $stmt = mysqli_prepare($link, 'INSERT INTO annuncio (titoloAnnuncio, descrizioneAnnuncio, idUtente) VALUES (?, ?, ?)');

    mysqli_stmt_bind_param($stmt, 'ssi', $titolo, $descrizione, $idCreatore);

    mysqli_stmt_execute($stmt);

    mysqli_stmt_close($stmt);

    disconnectFromDatabase($link);

}
```

Aggiungere un utente

```
function addUserToDatabase(array $valori){

    $link = connectToDatabase();



    //Controllo che l'email non sia già presente

    mysqli_query($link, 'SELECT * FROM datiAccount WHERE datiAccount.email = "'.$valori['email'].'"');
```

```
    if (mysqli_affected_rows($link) == 0) {

        mysqli_begin_transaction($link, MYSQLI_TRANS_START_READ_WRITE);

        //INSERISCO I DATI ANAGRAFICI

        $datiAnagrafici = 'INSERT INTO datiAnagrafici (nomeUtente, sesso, dataNascita, luogoNascita, luogoResidenza) VALUES (?,?,?,?,?)';

        $stmt = mysqli_prepare($link, $datiAnagrafici);

        mysqli_stmt_bind_param($stmt, 'sssss', $valori['nome_utente'], $valori['sesso'], $valori['data_n'], $valori['luogo_n'], $valori['luogo_r']);

        mysqli_stmt_execute($stmt);

        $idDatiAnag = mysqli_stmt_insert_id($stmt);

        //INSERISCO DATI ACCOUNT

        $datiAccount = 'INSERT INTO datiAccount(email, password, tipoUtente, numCartaCredito) VALUES (?,?,?,?)';

        $stmt_acc = mysqli_prepare($link, $datiAccount);

        mysqli_stmt_bind_param($stmt_acc, 'sssi', $valori['email'], $valori['pswd'], $valori['tipo'], $valori['num_carta']);

        mysqli_stmt_execute($stmt_acc);

        $idDatiAcc = mysqli_stmt_insert_id($stmt_acc);

        //INSERISCO DATI Carriera

        mysqli_query($link, 'INSERT INTO datiCarriera () VALUES ()');

        $idDatiCarr = mysqli_insert_id($link);

        //INSERISCO L'UTENTE

        $utente = 'INSERT INTO utente (idDatiAnagrafici, idDatiAccount, idDatiCarriera) VALUES (?, ?, ?)';

        $stmt_user = mysqli_prepare($link, $utente);

        mysqli_stmt_bind_param($stmt_user, 'iii', $idDatiAnag, $idDatiAcc, $idDatiCarr);

        mysqli_stmt_execute($stmt_user);

        $last_id = mysqli_stmt_insert_id($stmt_user);

        mysqli_commit($link);
```

```
            //considero l'utente come correntemente loggato al sito
            session_start();
            $_SESSION['sessione_user'] = $last_id;
        }
        else {
            die('Impossibile completare la registrazione!! E-mail già presente nel database');
        }
        disconnectFromDatabase($link);
}
```

Rimuovere una competenza lavorativa di un utente

```
function removeCompetenza($idCompetenza, $idCarriera){
        $link = connectToDatabase();
        mysqli_query($link, 'DELETE FROM carriera_has_competenza WHERE idCompetenza='.$idCompetenza.' AND idDatiCarriera='.$idCarriera);
        disconnectFromDatabase($link);
}
```

Aggiornare i dati account di un utente

```
function updateDatiAccount($valori){
        $link = connectToDatabase();
        mysqli_query($link, 'SELECT * FROM datiAccount WHERE datiAccount.email="'.$valori['email'].'" AND datiAccount.idDatiAccount != (SELECT u.idDatiAccount FROM utente AS u WHERE u.idUtente="'.getUserID().'")');
        if (mysqli_affected_rows($link) == 0){
            //l'email non è mai stata usata da nessuno
            $stmt = mysqli_prepare($link, 'UPDATE datiAccount AS acc SET acc.email=?, acc.password=?, acc.tipoUtente=?, acc.numCartaCredito=? WHERE acc.idDatiAccount = (SELECT u.idDatiAccount FROM utente AS u WHERE u.idUtente=?)');
            mysqli_stmt_bind_param($stmt, 'ssssi', $valori['email'], $valori['pswd'], $valori['tipo'], $valori['carta'], getUserID());
```

```
            mysqli_stmt_execute($stmt);

            mysqli_stmt_close($stmt);

            logout();

        }

        else{

            //l'email è utilizzata da qualcun'altro

            die('Email già utilizzata per un altro account');

        }

}
```

Aggiungere una valutazione

```
function valutaUtente($idValutato, $val){

        $link = connectToDatabase();

        $res = mysqli_query($link, 'SELECT val.idValutazione FROM valutazione AS val WHERE val.valutazione ="'.$val.'" AND val.idValutante='.getUserID());

        $idVal = mysqli_fetch_assoc($res);

        $idVal = $idVal['idValutazione'];

        if ($idVal == null){

            //non è presente la valutazione

            $stmt = mysqli_prepare($link, 'INSERT INTO valutazione (valutazione, idValutante) VALUES (?, ?)');

            mysqli_stmt_bind_param($stmt, 'si', $val, getUserID());

            mysqli_stmt_execute($stmt);

            $idVal = mysqli_stmt_insert_id($stmt);

            mysqli_stmt_close($stmt);

        }

        $stmt = mysqli_prepare($link, 'INSERT INTO utente_has_valutazione (idValutato, idValutazione) VALUES (?, ?)');

        mysqli_stmt_bind_param($stmt, 'ii', $idValutato, $idVal);
```

```
    mysqli_stmt_execute($stmt);

    mysqli_stmt_close($stmt);

    disconnectFromDatabase($link);

}
```

Rimuovere una richiesta di amicizia

```
function rimuoviRichiesta($link, $utenteDaAggiungere, $idUtente){

    //rimuovo le richieste di connessione che li coinvolgono

    $res = mysqli_query($link, 'SELECT ur.idUtente, rc.idRichiesta FROM richiestaConnessione AS rc INNER JOIN utente_has_richiesta AS ur ON rc.idRichiesta = ur.idRichiesta WHERE rc.idUtenteRichiedente=' . $utenteDaAggiungere . ' AND ur.idUtente=' . $idUtente);

    $richiesteConnesioni = array();

    while ($row = mysqli_fetch_assoc($res))

        $richiesteConnesioni[] = $row;

    foreach ($richiesteConnesioni as $rc) {

        mysqli_query($link, 'DELETE FROM richiestaConnessione WHERE idRichiesta=' . $rc['idRichiesta']);

        mysqli_query($link, 'DELETE FROM utente_has_richiesta WHERE idUtente=' . $rc['idUtente'] . ' AND idRichiesta=' . $rc['idRichiesta']);

    }

}
```

Note

NOTE

All'interno del file .zip oltre questa relazione sono stati inseriti vari file contenenti la struttura del database esportata e i sorgenti dell'applicazione client.

Per poter utilizzare al meglio l'intero progetto occorre seguire questi passaggi:

1. Aprire il proprio DBSM preferito ed importare il file di database esamedb.sql. (il database è stato realizzato su linux con mariadb e phpmyadmin); a seconda del tipo di DBSM la procedura potrebbe variare

2. Spostare tutti i file .php e il file .conf nella cartella del proprio web server oppure configurarlo in modo da utilizzare questa cartella come destinazione (durante lo sviluppo è stato utilizzato apache su linux)

3. Modificare il contenuto del file db.conf impostando l'hostname, l'username, la password e il nome del database con in valori corretti per permettere al web server di accedere al database correttamente (nel caso in cui dopo svariati tentativi il database continui a dare un errore di connessione si suggerisce di inserire i dati direttamente all'interno della funzione connectToDatabase() $link = new mysqli("hostane", "username", "password", "db");).

4. Connettersi da browser all'indirizzo del web server ed utilizzare il sito

Nota: al momento dell'installazione non saranno presenti alcuni dati dentro al database.