Esame di stato 2006 e-learning

Traccia

Una rete di scuole chiede che sia progettato e realizzato un database per l'organizzazione e la gestione del portale di una community di apprendimento sul Web.
L'organizzazione della community prevede che:

- l'accesso sia consentito ai soli utenti registrati;
- gli utenti siano distribuiti in tre gruppi: amministratore, docente, studente tali che:
  - un amministratore abbia accesso a tutte le aree protette del portale;
  - un docente abbia accesso a tutte le aree protette tranne che all'area di amministrazione;
  - uno studente abbia accesso alla propria area protetta e non abbia accesso né all'area di amministrazione né all'area riservata al gruppo docente;
- la registrazione degli utenti consenta:
  - alla rete di scuole di acquisire informazioni, sotto forma di dati non sensibili, relative agli utenti quali, ad esempio, nome e cognome, scuola o istituzione formativa di appartenenza,
  - collocazione geografica, e-mail, ecc….
  - agli utenti di scegliere un nome utente, una password e il gruppo di appartenenza tranne quello degli amministratori;
- agli utenti registrati, ciascuno per il proprio gruppo di appartenenza, sia consentito di effettuare l'upload di documenti multimediali archiviando:
  - il titolo
  - il tipo di documento (testo, audio, ecc.)
  - una descrizione sintetica
  - la data di upload
  - i dati personali che lo riguardano;
- il gruppo degli studenti possa usufruire di moduli formativi ad esso rivolti;
- il gruppo dei docenti possa usufruire sia di propri moduli formativi che di quelli rivolti agli studenti;
- ciascun modulo formativo sia individuato da un titolo, da una breve descrizione e dal tipo di utente cui è rivolto.

Il candidato, fatte eventuali ipotesi aggiuntive:

− fornisca:
  1. Lo schema concettuale e lo schema logico del database;
  2. La definizione delle relazioni in linguaggio SQL.

− implementi almeno una delle seguenti query:
  1. I docenti che hanno un account presso la community con la rispettiva collocazionegeografica ed i moduli formativi scelti.
  2. I dati relativi agli studenti e ai documenti che essi hanno inviato in remoto sul portale della community mediante upload.

− scriva, in un linguaggio lato server, il codice di almeno una delle seguenti pagine del portale:
  - con accesso riservato all'amministratore, il report che trae i dati dalla query n. 1
  - con accesso riservato agli studenti, il report che trae i dati dalla query n. 2
  - registrazione di un nuovo utente con eventuale invio automatico dei dati registrati mediante e-mail diretta all'utente appena registrato e ad un amministratore.

Analisi

Il problema proposto richiama alla mente sistemi web based molto diffusi nel campo educativo facenti parte della categoria dei LMS (Learning Management System). Si tratta di sistemi molto complessi basati su piattaforme di web attivo (web server/script interpreter/sql server) che forniscono tra l'altro portali con accesso autenticato, gestione dei ruoli, download e upload delle risorse.
Esempi molto diffusi di questi sistemi sono MOODLE, DOCEBO e CLAROLINE nel campo dell'open source, WEBCT e BLACKBOARD nel campo del software proprietario.
Non essendo possibile sviluppare un intero sistema LMS è necessario introdurre molte ipotesi aggiuntive che limitino l'estensione del problema.
La struttura che verrà descritta ricalca, con notevoli semplificazioni, il modello di una risorsa glossario dell'ambiente MOODLE.

Ipotesi aggiuntive

Accesso autenticato
Si prevede una autenticazione di sessione (server side) basata su cookies client side.
Alla prima apertura di una sessione il primo script che viene invocato registra un cookie di sessione nel browser, il quale successivamente lo trasmetterà a tutti gli script della stessa sessione.
Alla prima apertura di sessione viene anche effettuata la richiesta delle credenziali.
Le credenziali per l'accesso sono registrate nella banca dati stessa sotto forma di username e password cifrata con la codifica MD5.
Il superamento della verifica delle credenziali porta al mantenimento della sessione mediante la trasmissione da parte del client del cookie client side e da parte del server delle corrispondenti variabili di sessione server side che garantiscono la validità delle credenziali.
In questo modo dopo una prima autenticazione l'utente ha accesso alle altre pagine autenticate senza dovere rinnovare l'inserimento delle credenziali a patto di non chiudere il browser.
Il logout o la chiusura del browser porta alla rimozione delle variabili di sessione e quindi alla perdita delle credenziali.

Iscrizione al servizio
L'iscrizione, del tipo con conferma via e-mail, è gestita autonomamente dall'utente che compila una form con i dati.
Se i dati sono corretti l'iscrizione viene registrata come non confermata e viene inviata una mail all'indirizzo di posta specificato contenente un link ad una pagina di conferma della registrazione.
Alla conferma della registrazione viene mandata una email di conferma all'utente mentre non si inviano e-mail all'amministratore perché sono potenzialmente molteplici gli utenti che possono svolgere questo ruolo.
Una registrazione confermata è comunque una registrazione di tipo studente. In queste condizioni l'utente può autonomamente iscriversi a qualunque modulo riservato agli studenti e successivamente consultare le risorse di tali moduli o caricare risorse.
Il passaggio al ruolo di docente o amministratore deve essere effettuato da un amministratore. La prima parte di questa ultima ipotesi è in contrasto con la traccia che presuppone la possibilità di autoassegnarsi il ruolo di docente.
Un utente può appartenere solo ad un gruppo.

Dati non sensibili
Si suppone che l'accesso ai informazioni relative agli utenti sia possibile per tutti gli utenti registrati indipendentemente dal ruolo mentre questi dati non sono accessibili senza autenticazione.

Struttura delle risorse
Si suppone che le risorse condivise siano organizzate in moduli; la creazione dei moduli deve essere effettuata dagli amministratori mentre l'iscrizione ai moduli viene effettuata

autonomamente dall'utente. Gli studenti si possono iscrivere ai moduli per studenti; i docenti si possono iscrivere sia a moduli per studenti sia ai moduli per docenti; dal momento dell'iscrizione lo studente/docente ha la possibilità di vedere il contenuto del modulo a cui è iscritto ed ha la possibilità di caricare risorse multimediali nel modulo.
Non è previsto alcun sistema per la modifica delle risorse multimediali ne per la loro cancellazione.
Gli amministratori non si iscrivono ad alcun modulo ma hanno accesso a tutti i moduli e possono aggiornare e cancellare le risorse.

Upload delle risorse
L'upload delle risorse multimediali viene effettuato dall'utente autenticato attraverso una form di upload. La risorsa disponibile sul file system locale viene trasferita in un file temporaneo del server.
Lo script di upload registra la nuova risorsa nella banca dati e salva il file in una cartella del file system del server rinominandolo in base al valore della chiave primaria. L'archiviazione delle risorse multimediali è quindi effettuata con la tecnica "linked".
Lo svantaggio della tecnica linked rispetto alla tecnica embedded (risorsa inserita in un campo BLOB di una tabella del DB) è soprattutto dal punto di vista del backup che per le risorse deve essere fatto separatamente rispetto ai dati di banca dati.
La scelta "linked" è però giustificata dal fatto che le risorse multimediali possono essere di molti tipi diversi con potenziali problemi di archiviazione.

Individuazione delle entità.

Entità 'utente': è l'anagrafica degli utenti della community. Per accedere alle sezioni riservate del portale è necessario autenticarsi mediante un nome utente e password.
Queste due informazioni sono registrate per ogni utente, indipendentemente dal ruolo in questa entità. L'esigenza di identificare in modo univoco ogni utente porta alla definizione delle seguenti proprietà indispensabili:

- *username*:testo univoco scelto dall'utente che distingue ogni istanza e costituisce chiave primaria
- *password*:testo che memorizza in forma cifrata la password assegnata all'utente
- *cognome:*testo che contiene il cognome reale dell'utente
- *nome:* testo che contiene il nome reale dell'utente
- *email:* testo che contiene un indirizzo email valido dell'utente
- *confermato*: boolenano che indica se l'iscrizione è confermata
- *codice di conferma:* stringa casuale usata per la verifica della conferma

La necessità di aggregare informazioni in base alla zona di residenza e all'istituto di appartenenza porta alla normalizzazione di questi due attributi che diventano due tabelle di look-up associate 1:N con l'entità utente.
Ipotizzando che ogni utente possa svolgere nella community un singolo ruolo (studente,docente o amministratore) l'attriuto ruolo diventa una associazione 1:N con una tabella di look-up contenente i possibili ruoli.
I campi sono tutti obbligatori.
Poiché la chiave primaria non è ad autoincremento e viene scelta dall'utente è possibile che in fase di inserimento si verifichino collisioni che vanno risolte rifiutando l'inserimento e cambiando la chiave.

Entità 'comune': è una tabella di look-up contenente un insieme dei comuni del territorio di interesse della community. Ogni comune è identificato univocamente da una chiave artificiale a codice parlante definita da norme nazionali. Il codice è alfanumerico formato da un carattere alfabetico dalla 'A' alla 'Z' seguito da una stringa numerica di tre cifre (ad esempio Bologna è 'A944'). Ad ogni codice è associata una descrizione. I comuni sono ordinati alfabeticamente per codice. I campi sono tutti obbligatori

Entità 'istituto': è una tabella di look-up contenente un insieme degli istituti partecipanti alla community. Ogni istituto è identificato univocamente da una chiave artificiale a codice parlante definita dal ministero della pubblica istruzione norme nazionali. Il codice è alfanumerico

formato da 10 caratteri (ad esempio il codice del Belluzzi è 'BOTF030006'. Ad ogni codice è associata una descrizione. I campi sono tutti obbligatori.

Entità 'gruppo': è una tabella di look-up contenente tutti i possibili gruppi di appartenenza degli utenti. L'appartenenza ad un gruppo consente di definire i diritti concessi ad uno specifico utente. Ogni gruppo è identificato da una chiave primaria intera ad autoincremento e da una descrizione di tipo testo. I campi sono tutti obbligatori

Entità 'modulo': il modulo è l'entità principale che caratterizza i contenuti della community. Ogni modulo definisce un'area di condivisione di risorse consentendo sia la separazione delle risorse caricate che sono associate ad uno specifico modulo sia la separazione degli utenti che possono essere iscritti a nessuno, uno o più moduli.
Ogni modulo, che può essere creato solo dall'amministratore, è identificato univocamente da una chiave artificiale intera ad autoincremento ed ha i due seguenti attributi:

- *descrizione:* testo che descrive la natura del modulo
- *studente_docente:* booleano che distingue i moduli per docenti (accessibili ad amministratore e docenti) dai moduli per studenti (accessibili a tutti)

I campi sono tutti obbligatori.

Entità 'documento': il documento rappresenta una singola unità di contenuto delle risorse. Ogni documento appartiene ad un singolo modulo ed è stato caricato da un singolo utente.
Ogni documento, che può essere creato in modulo da un qualsiasi utente che abbia diritto di accesso sul modulo, è identificato univocamente da una chiave artificiale intera ad autoincremento ed ha i seguenti attributi:

- *titolo*: testo che descrive il titolo del documento
- *descrizione:* testo che riassume il contenuto del documento
- *data:* data di caricamento del documento

A ogni documento è associato un tipo di documento che ne definisce il formato di archiviazione; questa informazione e ottenuta mediante una associazione ad una tabella di look-up che contiene tutti i possibili tipi di documento.
A ogni documento è associato all'utente che lo ha caricato.
Il documento multimediale vero e proprio non viene archiviato nella istanza di questa entità ma viene copiato dopo l'upload in una cartella di sistema predefinita rinominato con un nome costruito in base alla chiave primaria assegnata al documento. In questo modo si realizza una archiviazione linked.

Entità 'tipo': è una tabella di look-up contenente tutti i possibili tipi di documenti multimediali. Ogni tipo di documento è identificato da una chiave primaria intera ad autoincremento e da una descrizione di tipo testo. I campi sono tutti obbligatori

Individuazione delle associazioni.

Associazione utente/comune 'risiede': ogni utente deve essere associato ad un comune di residenza selezionato dall'elenco di lookup dei comuni del territorio di interesse della community. Esiste quindi una associazione 1:N tra comune e utente. La relazione è parziale dal lato comune (un comune può non avere alcun utente residente) e totale dal lato utente (un utente deve risiedere in un comune)

Associazione utente/istituto 'appartiene': ogni utente deve essere associato ad un istituto di appartenenza selezionato dall'elenco di lookup degli istituti appartenenti alla community. Esiste quindi una associazione 1:N tra istituto e utente. La relazione è parziale dal lato istituto (un istituto può non avere alcun utente nella community) e totale dal lato utente (un utente deve appartenere ad un istituto della community)

Associazione utente/gruppo 'diritti': ogni utente deve essere associato ad un gruppo che ne definisce i diritti nell'ambito delle risorse della community. Il gruppo di appartenenza selezionato dall'elenco di lookup dei gruppi della community. Si ipotizza che un utente possa appartenere ad un solo gruppo e quindi abbia lo stesso tipo di diritti su tutti i moduli. Esiste

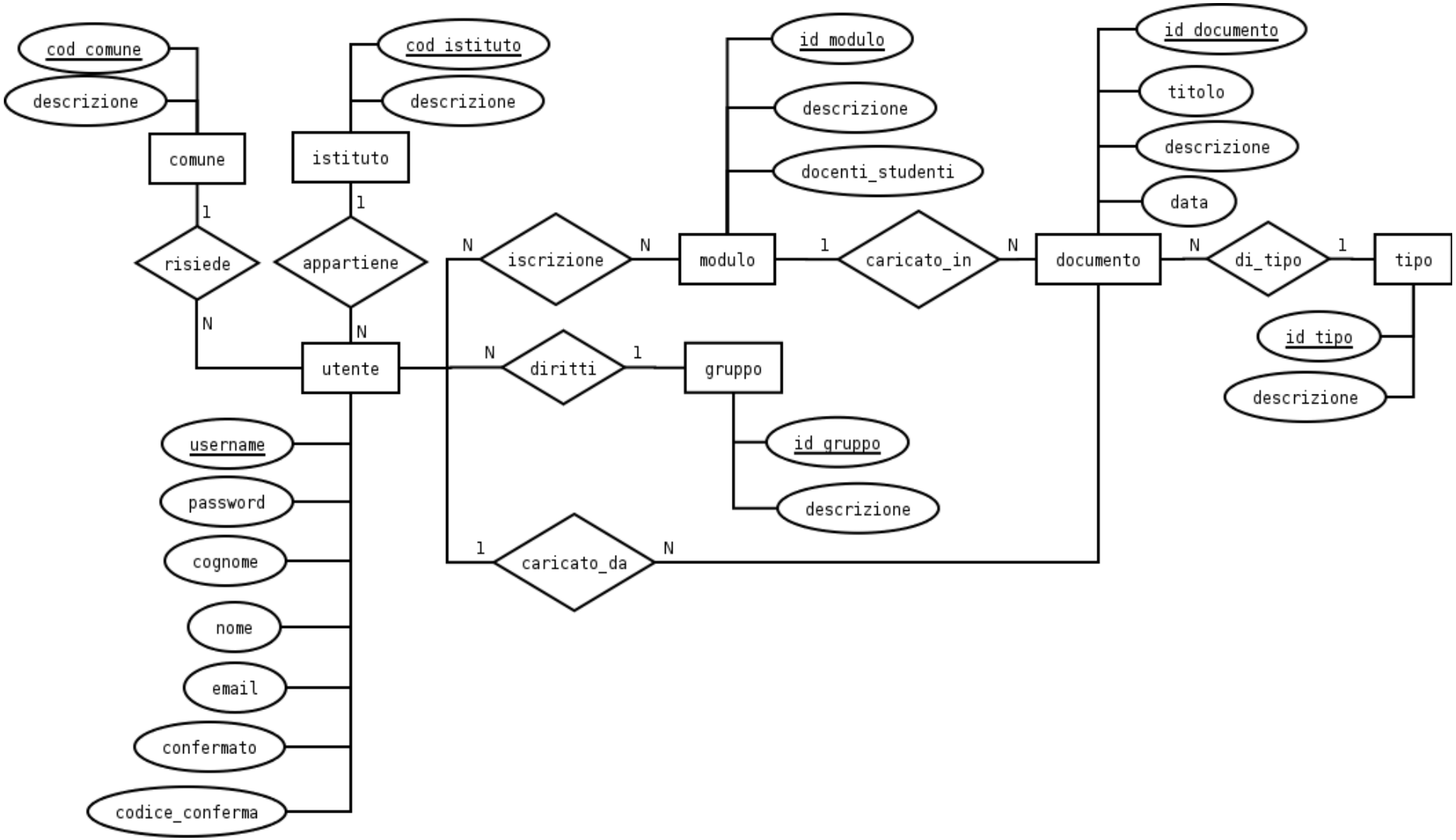
quindi una associazione 1:N tra gruppo e utente. La relazione è parziale dal lato gruppo (un gruppo può non avere alcun utente nella community) e totale dal lato utente (un utente deve appartenere ad un gruppo della community)

Associazione utente/modulo 'iscrizioni': ogni utente può essere iscritto a nessuno, uno o più moduli compatibilmente con i diritti definiti dal suo gruppo. Gli studenti possono essere iscritti solo ai moduli riservati agli studenti, i docenti possono essere iscritti sia ai moduli degli studenti sia ai moduli dei docenti, gli amministratori non sono iscritti ad alcun modulo ma hanno accesso a tutti. Ogni modulo può avere nessuno, uno o più utenti iscritti, ogni utente può essere iscritto a nessuno uno o più moduli. Esiste quindi una associazione N:N tra utenti e moduli. La relazione è parziale da entrambi i lati

Associazione modulo/documento 'caricato_in': ogni documento deve essere associato ad un modulo di appartenenza selezionato tra i moduli esistenti. Esiste quindi una associazione 1:N tra modulo e documento. La relazione è parziale dal lato modulo (un modulo può non avere alcun documento caricato) e totale dal lato documento (un documento deve essere caricato in un modulo)
Associazione utente/documento 'caricato_da': ogni documento deve essere associato ad un utente che lo ha caricato selezionato tra gli utenti esistenti. Esiste quindi una associazione 1:N tra utente e documento. La relazione è parziale dal lato utente (un utente può non avere caricato alcun documento) e totale dal lato documento (un documento deve essere stato caricato da un utente)

Associazione tipo/documento 'di_tipo': ogni documento deve essere associato ad un tipo di documento selezionato tra i tipi esistenti. Esiste quindi una associazione 1:N tra tipo e documento. La relazione è parziale dal lato tipo (un tipo può non corrispondere ad alcun documento caricato9 e totale dal lato documento (un documento deve corrispondere ad un tipo esistente)

Schema concettuale (modello ER)



Schema logico
Si usa il modello relazionale. Ogni entità ed entità debole viene sostituita con una relazione. Ogni associazione 1:N tra due entità viene sostituita con l'esportazione della chiave primaria del lato 1 come chiave esterna nel lato N. Ogni associazione N:N tra due entità viene sostituita con la generazione di una nuova relazione ottenuta esportando le due chiavi primarie come chiavi esterne e definendo la loro composizione come chiave primaria della nuova relazione.
Poiché il progetto non richiede il modello fisico nello schema logico vengono anche indicati i tipi di dato.

Relazioni
utente(username:testo,
password: testo,
cognome: testo,
nome: testo,
email: testo,
confermato: booleano,
codice_conferma: testo,
cod_comune: testo
cod_istituto: testo
id_gruppo: intero)
comune(cod_comune:testo,
descrizione:testo)
istituto(cod_istituto:testo,
descrizione: testo)
gruppo(id_gruppo:intero,
descrizione: testo)
modulo(id_modulo:intero,
descrizione:testo,
docenti_studenti: booleano)
iscrizione(username:testo,
id_modulo:intero)
tipo(id_tipo:intero,
descrizione: testo)
documento(id_documento:intero,
titolo: testo,
descrizione: testo,
data: data
id_tipo: intero,
id_modulo: intero,
username: testo)

Vincoli di integrità

Vincoli di integrità referenziale

utente.cod_comune $\subseteq$ comune.cod_comune
utente.cod_istituto $\subseteq$ istituto.cod_comune
utente.id_gruppo $\subseteq$ gruppo.id_gruppo
iscrizione.username $\subseteq$ utente.username
iscrizione.id_modulo $\subseteq$ iscrizione.id_modulo
documento.id_tipo $\subseteq$ tipo.id_tipo
documento.id_modulo $\subseteq$ modulo.id_modulo
documento.username $\subseteq$ utente.username

Modello fisico

```
CREATE TABLE utente (
  username varchar(20) NOT NULL default '',
  password varchar(32) NOT NULL default '',
  cognome varchar(80) NOT NULL default '',
  nome varchar(80) NOT NULL default '',
  email varchar(255) NOT NULL default '',
  cod_comune varchar(4) NOT NULL default '',
  cod_istituto varchar(10) NOT NULL default '',
  id_gruppo int(11) NOT NULL default '0',
  confermato tinyint(1) NOT NULL default '0',
  codice_conferma varchar(32) NOT NULL default '',
  PRIMARY KEY  (username),
  KEY cod_comune (cod_comune),
  KEY cod_istituto (cod_istituto),
  KEY id_gruppo (id_gruppo)
  FOREIGN KEY (cod_comune) REFERENCES comune (cod_comune),
  FOREIGN KEY (cod_istituto) REFERENCES istituto (cod_istituto),
  FOREIGN KEY (id_gruppo) REFERENCES gruppo (id_gruppo)
);

CREATE TABLE comune (
  cod_comune varchar(4) NOT NULL default '',
  descrizione varchar(80) NOT NULL default '',
  PRIMARY KEY  (cod_comune)
);

CREATE TABLE gruppo (
  id_gruppo int(11) NOT NULL auto_increment,
  descrizione varchar(20) NOT NULL default '',
  PRIMARY KEY  (id_gruppo)
);

CREATE TABLE istituto (
  cod_istituto varchar(10) NOT NULL default '',
  descrizione varchar(80) NOT NULL default '',
  PRIMARY KEY  (cod_istituto)
);

CREATE TABLE modulo (
  id_modulo int(11) NOT NULL auto_increment,
  descrizione varchar(80) NOT NULL default '',
  docente_studente tinyint(1) NOT NULL default '0',
  PRIMARY KEY  (id_modulo)
);
```

```
CREATE TABLE iscrizione (
  username varchar(20) NOT NULL default '',
  id_modulo int(11) NOT NULL default '0',
  PRIMARY KEY  (username,id_modulo),
  KEY id_modulo (id_modulo)
  FOREIGN KEY (username) REFERENCES utente (username),
  FOREIGN KEY (id_modulo) REFERENCES modulo (id_modulo);
);

CREATE TABLE tipo (
  id_tipo int(11) NOT NULL auto_increment,
  descrizione varchar(80) NOT NULL default '',
  PRIMARY KEY  (id_tipo)
);

CREATE TABLE documento (
  id_documento int(11) NOT NULL auto_increment,
  titolo varchar(80) NOT NULL default '',
  descrizione text NOT NULL,
  data date NOT NULL default '0000-00-00',
  id_tipo int(11) NOT NULL default '0',
  id_modulo int(11) NOT NULL default '0',
  username varchar(20) NOT NULL default '',
  PRIMARY KEY  (id_documento),
  KEY id_tipo (id_tipo),
  KEY id_modulo (id_modulo),
  KEY username (username)
  FOREIGN KEY (id_tipo) REFERENCES tipo (id_tipo),
  FOREIGN KEY (id_modulo) REFERENCES modulo (id_modulo);
  FOREIGN KEY (`username`) REFERENCES `utente` (`username`)
);
```

Interrogazioni

n. 1: I docenti che hanno un account presso la community con la rispettiva collocazione geografica ed i moduli formativi scelti.

Si effettua una giunzione naturale sulla associazione N:N utente-iscrizione-modulo per estrarre gli utenti iscritti ai moduli.
Le due ulteriori giunzioni sulle associazioni 1:N con comune e gruppo servono rispettivamente per estrarre la zona ed il ruolo.
Una restrizione sulla descrizione del ruolo estrae gli utenti che appartengono al ruolo 'docente'.
La proiezione estrae i dati anagrafici degli utenti, il comune di residenza ed i moduli a cui sono iscritti.
L'ordinamento mostra i gli utenti in ordine alfabeto crescente.

```
SELECT    t1.cognome,
          t1.nome,
          t3.descrizione AS comune,
          t5.descrizione AS modulo,
          t5.docente_studente AS docente
FROM      utente AS t1,
          gruppo AS t2,
          comune AS t3,
          iscrizione AS t4,
          modulo AS t5
WHERE     t1.id_gruppo=t2.id_gruppo
AND       t1.cod_comune=t3.cod_comune
AND       t1.username=t4.username
AND       t4.id_modulo=t5.id_modulo
AND       t2.descrizione= 'docente'
ORDER BY  t1.cognome,t1.nome,t5.descrizione
```

n. 2: I dati relativi agli studenti e ai documenti che essi hanno inviato in remoto sul portale della community mediante upload.

Si effettua una giunzione naturale tra utente, gruppo, documento,, modulo e tipo. L'associazione iscrizione tra modulo e utente è esclusa dal join perché l'informazione di iscrizione al modulo si ottiene indirettamente dall'associazione 1:N tra utente e documento e dall'associazione 1:N tra documento e modulo.

Le due ulteriori giunzioni sulle associazioni 1:N con tipo e gruppo servono rispettivamente per estrarre il tipo di documento ed il ruolo dell'utente.

Una restrizione sulla descrizione del ruolo estrae gli utenti che appartengono al ruolo 'studente'.

La proiezione estrae i dati anagrafici degli utenti, i dati del documento caricato compreso il modulo di appartenenza.

L'ordinamento mostra i gli utenti in ordine alfabeto crescente.

```
SELECT    t1.username,
          t1.cognome,
          t1.nome,
          t4.descrizione AS modulo,
          t3.titolo,
          t3.descrizione AS riassunto,
          t5.descrizione AS tipo,
          t3.data
FROM      utente AS t1,
          gruppo AS t2,
          documento AS t3,
          modulo AS t4,
          tipo AS t5
WHERE     t1.id_gruppo=t2.id_gruppo
AND       t1.username=t3.username
AND       t3.id_modulo=t4.id_modulo
AND       t3.id_tipo=t5.id_tipo
AND       t2.descrizione= 'studente'
ORDER BY  t1.cognome,t1.nome,t4.descrizione,t3.titolo
```

Premesse comuni per tutti gli script lato server

Ambiente operativo

Si ipotizza di usare una piattaforma xAMP composta da:

- x sistema operativo Windows o Linux
- A web server Apache
- M sql server MySQL
- P script interprete PHP

Connessione al database

Il server MySQL si presenta come un server TCP che risponde sulla porta 3306.
L'interpreter PHP dispone di una libreria di funzioni di interfaccia con il server MySQL.
Si può quindi realizzare uno script di connessione da includere in ogni script che necessiti dell'accesso alla banca dati (connect.php).

```
<?php
 //connette al SQL server sullo stesso host,
 //utente nobody, password 'qwerty'
 $sock=mysql_connect('localhost','nobody','qwerty');
 //se la connessione fallisce termina segnalando l'errore
 if ($sock==0) die(mysql_error());
 //seleziona il database sulla connessione
 $ris=mysql_select_db('community',$sock);
 //se la selezione fallisce termina segnalando l'errore
 if ($ris==0) die(mysql_error());
?>
```

La connessione avviene da parte dell'agente (utente fittizio) 'nobody'. Affinché la connessione abbia successo è necessario che l'amministratore del database garantisca l'accesso all'utente nobody con il seguente DCL:

```
GRANT SELECT,INSERT,UPDATE,DELETE
      ON community.*
      TO nobody@localhost
      IDENTIFIED BY 'qwerty'
```

Per motivi di sicurezza l'accesso alla banca dati è limitato agli script che si trovano sullo stesso host dell'sql server (localhost)

Gli script che necessitano di una connessione devono includere lo script di connessione:

```
<?php
 //termina se l'inclusione fallisce
 require 'connect.php'
?>
```

Autenticazione

Si ipotizza di realizzare una autenticazione di sessione PHP.

L'autenticazione di sessione si basa sulla creazione di una sessione all'avvio di ogni script:

```
<?php session_start(); ?>
```

L'avvio di una sessione rende disponibili allo script le variabili di sessione lato server contenute nell'array $_SESSION[]. La prima volta che viene lanciato uno script di sessione il server forza la registrazione di un cookie di sessione nel client (i cookies devono essere abilitati per l'host che fa la richiesta). Il client mantiene il cookie di sessione fino alla rimozione da parte del server o fino alla chiusura del browser e ogni volta che richiede una pagina via GET o POST invia anche il cookie che consente al server di estrarre le corrispondenti variabili lato server ($_SESSION[]). In questo modo due pagine di una stessa sessione condividono le stesse variabili globali. L'autenticazione di sessione si basa sulla verifica dell'esistenza di una variabile di sessione che identifica l'utente. Supponiamo di chiamare 'username' la chiave associativa dell'elemento dell'array $_SESSION ($_SESSION['username']) che identifica l'utente e supponiamo che ad autenticazione superata contenga il nome dell'utente. Ogni pagina che deve essere sottoposta ad autenticazione deve contenere il seguente script (session.php):

```
<?php
 //crea una sessione o riprende una sessione già aperta
 session_start();
 //verifica se già autenticato
 if (!isset($_SESSION['username'])) { //non esiste 'username'
  //non autenticato: redireziona alla form di login
  header("location: login.php");
}
//se arrivo qui vuole dire che sono autenticato: mostro la pagina che mi include
?>
```

Se un'altra pagina della stessa sessione ha fatto l'autenticazione lo script non fa nulla altrimenti redireziona alla form di login. La form di login raccoglie l'username e la password (in chiaro) e le trasferisce ad uno script di verifica delle credenziali. Supponendo che le credenziali ricevute dalla form sia contenute nelle variabili $username e $password il seguente script controlla la validità delle credenziali:

```
<?php
 //estrae dalla banca dati un eventuale utente con queste credenziali
 $msg="SELECT CONCAT(t1.nome,' ',t1.cognome) AS utente,t2.descrizione AS ruolo
       FROM utente AS t1,gruppo AS t2
       WHERE t1.id_gruppo=t2.id_gruppo
       AND username='$username'
       AND password=MD5('$password')
       AND confermato=1";
 $query=mysql_query($msg,$sock);
 if ($row_user=mysql_fetch_assoc($query)) { //trovato accetta credenziali
  //crea le variabili di autenticazione di sessione
  $_SESSION['username']=$username;
  $_SESSION['utente']=$row_user['utente'];
  $_SESSION['ruolo']=$row_user['ruolo'];
 }
 else {  //credenziali rifiutate: rilancia la form di login
  header("Location: login.php");
 }
?>
```

La selezione delle credenziali dalla banca dati è una ricerca per chiave primaria (username) quindi produce 0 oppure 1 record in uscita; se si ottiene un record ($row_user non nullo) i valori di username, cognome+nome e ruolo vengono usati come variabili di sessione. Il primo serve per verificare se l'autenticazione ha avuto successo e chi è l'utente autenticato, il secondo serve per segnalare all'utente la permanenza in area autenticata ed il terzo serve per personalizzare le pagine in base al ruolo senza dovere reinterrogare la BD ogni volta.

Se $row_user è nullo l'autenticazione è fallita (non si trova la coppia username e cifratura della password in banca dati) quindi si torna alla form di autenticazione mediante una redirezione.

Gli script che necessitano di autenticazione devono includere lo script di sessione:

```
<?php  require 'session.php'  //verif.sessione ed event. lancia login ?>
```

Con accesso riservato all'amministratore, il report che trae i dati dalla query n. 1
Lo script effettua prima di tutto la connessione. Se la connessione ha successo effettua la verifica dello stato di autenticazione. Se l'autenticazione ha successo verifica che l'utente autenticato appartenga al gruppo degli amministratori e solo in questo caso viene effettuata la query di selezione già presentata come query n.1. Se la query ha successo viene presentata la pagina. Nella prima riga vengono mostrati i principali dati dell'utente autenticato (username, nome+cognome, diritti). Viene poi presentata una tabella dinamica. Il numero di colonne corrisponde alla proiezione della query, le righe vengono generate dinamicamente percorrendo la cardinalità del risultato della query. I campi di testo vengono filtrati per eliminare i caratteri di escape.La spunta di modulo per docenti viene generata con l'operatore ternario : *(<condizione> ?<espr.vero> :<espr.falso>)*

```
<?php require 'connect.php' //connessione alla banca dati ?>
<?php require 'session.php' //verifica stato di autenticazione ?>
<?php //accesso consentito solo agli amministratori
 if($_SESSION['ruolo']!='amministratore') die('riservato agli amministratori');
 $msg="SELECT     t1.cognome, //estrae dati da bd
                  t1.nome,
                  t3.descrizione AS comune,
                  t5.descrizione AS modulo,
                  t5.docente_studente AS docente
      FROM        utente AS t1,
                  gruppo AS t2,
                  comune AS t3,
                  iscrizione AS t4,
                  modulo AS t5
      WHERE       t1.id_gruppo=t2.id_gruppo
      AND         t1.cod_comune=t3.cod_comune
      AND         t1.username=t4.username
      AND         t4.id_modulo=t5.id_modulo
      AND         t2.descrizione= 'docente'
      ORDER BY    t1.cognome,t1.nome,t5.descrizione";
 $query=mysql_query($msg,$sock);
 if($query==0) die(mysql_error());
?>
<html>
<head><title>query n.1</title></head>
<body>
<?php  //mostra dati utente autenticato
 echo $_SESSION['username']."-".$_SESSION['utente']."-".$_SESSION['ruolo']
?>
<table>
  <tr>      <!-- riga statica di intestazione -->
    <td>cognome</td>
    <td>nome</td>
    <td>comune</td>
    <td>modulo</td>
    <td>solo docenti </td>
  </tr>
<?php while($row_user=mysql_fetch_assoc($query)) { //riga dinamica ?>
  <tr>
    <td><?php echo stripslashes($row_user['cognome']) ?></td>
    <td><?php echo stripslashes($row_user['nome']) ?></td>
    <td><?php echo stripslashes($row_user['comune']) ?></td>
    <td><?php echo stripslashes($row_user['modulo']) ?></td>
    <td><?php echo (($row_user['docente']!=0)?'X':' ') ?></td>
  </tr>
<?php } ?>
</table>
</body>
</html>
```

Con accesso riservato agli studenti, il report che trae i dati dalla query n. 2
L'algoritmo di gestione di questa seconda pagina è analogo a quello della precedente.

```
<?php require 'connect.php' //connessione alla banca dati ?>
<?php require 'session.php' //verifica stato di autenticazione ?>
<?php //accesso consentito a tutti
 $msg="SELECT     t1.username,
                  t1.cognome,
                  t1.nome,
                  t4.descrizione AS modulo,
                  t3.titolo,
                  t3.descrizione AS riassunto,
                  t5.descrizione AS tipo,
                  t3.data
      FROM        utente AS t1,
                  gruppo AS t2,
                  documento AS t3,
                  modulo AS t4,
                  tipo AS t5
      WHERE       t1.id_gruppo=t2.id_gruppo
      AND         t1.username=t3.username
      AND         t3.id_modulo=t4.id_modulo
      AND         t3.id_tipo=t5.id_tipo
      AND         t2.descrizione= 'studente'
      ORDER BY    t1.cognome,t1.nome,t4.descrizione,t3.titolo";
 $query=mysql_query($msg,$sock);
?>
<html>
<head><title>query n.2</title></head>
<body>
<?php  //mostra dati utente autenticato
 echo $_SESSION['username']."-".$_SESSION['utente']."-".$_SESSION['ruolo']
?>
<table>
  <tr>      <!-- riga statica di intestazione -->
    <td>username</td>
    <td>cognome</td>
    <td>modulo</td>
    <td>titolo</td>
    <td>riassunto</td>
    <td>tipo</td>
    <td>data</td>
  </tr>
<?php while($row_user=mysql_fetch_assoc($query)) { //riga dinamica ?>
  <tr>
    <td><?php echo $row_user['username'] ?></td>
    <td><?php echo stripslashes($row_user['cognome']) ?></td>
    <td><?php echo stripslashes($row_user['nome']) ?></td>
    <td><?php echo stripslashes($row_user['modulo']) ?></td>
    <td><?php echo substr(stripslashes($row_user['riassunto']),0,80) ?></td>
    <td><?php echo stripslashes($row_user['tipo']) ?></td>
    <td><?php echo $row_user['data'] ?></td>
  </tr>
<?php } ?>
</table>
</body>
</html>
```

Registrazione di un nuovo utente con eventuale invio automatico dei dati registrati mediante e-mail diretta all'utente appena registrato e ad un amministratore.

Il sistema di registrazione di un nuovo utente si basa sulla conferma via email. Questa tecnica consente di combinare una gestione automatizzata delle registrazioni con le esigenze di sicurezza evitando registrazioni false.
La form di registrazione di un nuovo utente chiede l'inserimento dei seguenti dati obbligatori:

- *username*: una stringa di max 20 caratteri
- *password*: una stringa di max 32 caratteri
- *ripetizione password*: la stessa stringa del campo precedente
- *cognome*: una stringa di max 80 caratteri
- *nome*: una stringa di max 80 caratteri
- *email*: una stringa di max 255 caratteri
- *selezione di un comune di residenza*: un comune estratto dalla look-up dei comuni
- *selezione di un istituto di appartenenza*: un istituto estratto dalla look-up degli isti.

Un controllo client side verifica che nessuno dei campi sia vuoto o non selezionato, che username sia in un formato valido e che i due inserimento di password coincidano.
La form viene sottomessa al server. Lo script di risposta effettua le seguenti operazioni:

- *verifica della collisione di username:* controlla la non esistenza dell'username scelto (che è chiave primaria nella tabella utente); se esiste redireziona con fallimento alla form di registrazione
- *registrazione non confermata dell'utente:*se l'username è valido inserisce il nuovo utente in banca dati con il campo confermato=0 ed una stringa generata casualmente nel codice_conferma
- *invio di una email all'utente:*la mail contiene l'URL dello script di conferma con due parametri di GET:
    - *username:* stringa contenente l'username da confermare
    - *codice-conferma* stringa contenente lil codice di conferma archiviato
- *invito a confermare l'iscrizione:* si chiede all'utente di leggere il messaggio ricevuto via email e di visitare l?URL che viene proposto nella mail

In questo modo l'utente è già registrato in banca dati ma non è operativo perché non è confermato.
Se l'utente riceve il messaggio di posta e visita l'URL proposto l'iscrizione viene confermata e l'utente diventa operativo. Vengono anche inviati due messaggi di posta: uno all'utente per la conferma della attivazione e una notifica agli amministratori.

Richiesta di registrazione (richiesta_registrazione.php)
E' un modulo di richiesta statico che sottomette lo script di risposta.
La validità dei dati è controllata da uno script client side.

```
<?php //connessione ed estrazione delle tabelle di look-up
 require 'connect.php'
 $msg="SELECT * FROM comune";
 $qcomune=mysql_query($msg,$sock);
 $msg="SELECT * FROM istituto";
 $qist=mysql_query($msg,$sock);
?>
<html>
<head><title>registrazione nuovo utente</title></head>
<script type="text/javascript" src="./convalida.js"></script>
<body>
registrazione nuovo utente
<form action="risposta_registrazione.php" method="POST" name="registra"
onsubmit="return convalida(this)">
username
<input name="username" type="text">
password
<input name="password" type="password">
ripeti password
<input name="repeat" type="password">
cognome
<input name="cognome" type="text" >
nome
<input name="nome" type="text" >
email
<input name="email" type="text">
comune <!-- casella a discesa di selezione del comune -->
<select name="cod_comune" id="cod_comune">
<option value="0000">selezionare un comune</option>
<?php while($row_comune=mysql_fetch_assoc($qcomune)) { ?>
<option value="<?php echo $row_comune['cod_comune'] ?>"><?php echo
$row_comune['descrizione'] ?></option>
<?php } ?>
</select>
istituto <!-- casella a discesa di selezione dell'istituto -->
<select name="cod_istituto" id="cod_istituto">
<option value="0000000000">selezionare un istituto</option>
<?php while($row_ist=mysql_fetch_assoc($qist)) { ?>
<option value="<?php echo $row_ist['cod_istituto'] ?>"><?php echo
$row_ist['descrizione'] ?></option>
<?php } ?>
</select>
<input type="submit" name="Submit" value="Invia" >
 </form>
</body>
</html>
```

Convalida client side (convalida.js)
Lo script controlla la nullità dei campi di testo, la diversità tra la password e la ripetizione e la mancata selezione delle caselle a discesa.

```
function convalida(miaform) {
 if(miaform.username.value=="") {
  alert("manca username");
  return false;
 }
 if(miaform.password.value=="") {
  alert("manca password");
  return false;
 }
 if(miaform.password.value!==miaform.repeat.value) {
  alert("la ripetizione password non corrisponde");
  return false;
 }
 if(miaform.cognome.value=="") {
  alert("manca cognome");
  return false;
 }
 if(miaform.nome.value=="") {
  alert("manca nome");
  return false;
 }
 if(miaform.email.value=="") {
  alert("manca email");
  return false;
 }
 if(miaform.cod_comune.options[miaform.cod_comune.selectedIndex].value=="0000")
{
  alert("Selezionare il comune di residenza");
  return false;
 }
 if(miaform.cod_istituto.options[miaform.cod_istituto.selectedIndex].value
=="0000000000") {
  alert("Selezionare l'istituto");
  return false;
 }
 return true;
}
```

Risposta di registrazione (risposta_registrazione.php)
E' lo script di risposta alla sottomissione; lo script controlla l'esistenza dei parametri, genera una stringa casuale di conferma, inserisce i dati del nuovo utente in banca dati come utente non confermato. In caso di duplicazione della chiave primaria l'inserimento fallisce e l'utente deve fare una nuova registrazione con un diverso nome utente. In caso si successo viene inviata all'indirizzo di posta specificato una mail contenente l'URL di conferma della registrazione con l'username ed il codice di conferma passato come parametro di GET.

```
<?php //verifica l'esistenza dei parametri di ingresso
 if (!isset($_POST['username'])) die('manca username');
 else $username=$_POST['username'];
 ...
 if (!isset($_POST['cod_istituto'])) die('manca istituto');
 else $cod_istituto=$_POST['cod_istituto'];
 //genera un codice casuale di conferma
 $codice_conferma=bin2hex(rand(-1E+18,+1E+18));
 require 'connect.php';
 $msg="INSERT INTO utente
            (username,password,cognome,nome,email,
             cod_comune,cod_istituto,id_gruppo,
             confermato,
             codice_conferma)
          VALUES(
             '$username',MD5('$password'),'$cognome','$nome','$email',
             '$cod_comune','$cod_istituto',3, //studente
             0, //non confermato
             '$codice_conferma')";
 $query=mysql_query($msg,$sock);
 //in caso di duplicazione della chiave torna all'inserimento
 if($query==0) header("location: richiesta_registrazione.php");
 //invia mail all'indirizzo fornito
 $subject="richiesta registrazione community";
 $mailmsg=
"Caro $nome $cognome,\r\nPer completare la registrazione percorri il seguente
link\r\n\http://studenti.scuole.bo.it/.../conferma_registrazione.php?username=$u
sername&codice_conferma=$codice_conferma\r\nCordiali Saluti,\r\nl'amministratore
della community\r\n";
 $headers="From: webmaster@community\r\nReply-To: webmaster@communty\r\n";
 @mail($email,$subject,$mailmsg,$headers);
?>
<html>
<head>
<title>risposta registrazione</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
risposta registrazione
Grazie per richiesta la registrazione <b><?php echo $nome." ".$cognome
?></b><br>
Una mail contenente le istruzioni per completare la registrazione è stata
mandata all'indirizzo <b><?php echo $email ?></b>.<br>
A presto!
</body>
</html>
```

Conferma di registrazione (conferma_registrazione.php)

E' lo script di conferma della registrazione; lo script viene lanciato direttamente dalla mail ricevuta dall'utente dove è presente con i suoi parametri di GET (username e codice di conferma)

```
<?php  //verifica l'esistenza dei parametri di ingresso
 if (!isset($_GET['username'])) die('manca username');
 else $username=$_GET['username'];
 if (!isset($_GET['codice_conferma'])) die('manca codice conferma');
 else $codice_conferma=$_GET['codice_conferma'];
 //cerca utente in banca dati
 require 'connect.php';
 $msg="SELECT *
       FROM utente
       WHERE username='$username'
       AND codice_conferma='$codice_conferma'";
 $query=mysql_query($msg,$sock);
 if($query==0) die(mysql_error());
 //se i dati non sono corretti abbandona
 if (mysql_num_rows($query)==0) die('conferma di registrazione non valida');
 //se i dati sono corretti estrae le informazioni sull'utente
 $row_utente=mysql_fetch_assoc($query);
 $nome=$row_utente['nome'];
 $cognome=$row_utente['cognome'];
 $email=$row_utente['email'];
 //convalida l'utente
 $msg="UPDATE utente SET confermato=1 WHERE username='$username'";
 $query=mysql_query($msg,$sock);
 if($query==0) die(mysql_error());
 //invia un email all'indirizzo specificato
 $subject="conferma registrazione community";
 $mailmsg=
"Caro $nome $cognome,\r\nLa tua registrazione è stata completata\r\nCordiali
Saluti,\r\nl'amministratore della community\n";
 $headers="From: webmaster@community\r\nReply-To: webmaster@communty\r\n";
 @mail($email,$subject,$mailmsg,$headers);
?>
<html>
<head><title>Conferma registrazione</title></head>
<body>
<p>Grazie per la registrazione <b><?php echo $nome." ".$cognome ?></b><br>
Ora puoi accedere alle aree autenticate usando la tua username e password.</p>
<p><a href="index.php">indietro</a><br>
</p>
</body>
</html>
```