A038 - ESAME DI STATO CONCLUSIVO DEL SECONDO CICLO DI ISTRUZIONE
Indirizzo ITIA - INFORMATICA E TELECOMUNICAZIONI ARTICOLAZIONE "INFORMATICA"

Disciplina: INFORMATICA

Il candidato svolga la prima parte della prova e due tra i quesiti proposti nella seconda parte.

PRIMA PARTE

Una scuola vuole progettare una piattaforma web per la fruizione di *educational games* (ovvero videogiochi in ambito educativo), per migliorare l'apprendimento nelle varie materie.
Ciascun docente, una volta completata la registrazione alla piattaforma, può creare una o più classi virtuali (identificate da un nome e una materia di pertinenza: es. 3B, matematica) e aprire l'iscrizione alle singole classi ai propri studenti tramite la condivisione del codice iscrizione (link o QR-code).
Nella piattaforma è presente il catalogo dei videogiochi didattici, classificati in base ad un elenco di argomenti prestabiliti (es: triangoli, legge di Ohm, verismo …): ciascun docente può selezionare uno o più videogiochi per includerli in una classe virtuale. Per ogni videogioco è presente un titolo, una descrizione breve di massimo 160 caratteri, una descrizione estesa, il numero di "monete virtuali" che si possono raccogliere all'interno del gioco e fino a tre immagini sul gioco.
Uno studente si iscriverà sulla piattaforma alle classi cui è stato invitato (es: 3B matematica, 3B italiano ...) tramite il relativo codice iscrizione, e all'interno di ciascuna classe troverà i link ai videogiochi didattici proposti dal docente. Svolgendo ciascun videogioco, lo studente potrà raccogliere sequenzialmente delle monete tramite quiz o attività da completare. Una moneta è un riconoscimento che viene assegnato nel videogioco al raggiungimento di determinati traguardi educativi graduali.
Attraverso il numero di monete, raccolte man mano da uno studente in ciascun videogioco di quella classe, si può determinare una classifica per ciascun gioco e anche una classifica generale comprensiva di tutti i giochi della classe; il docente può quindi seguire l'andamento degli studenti e supportarli individualmente nel completamento della raccolta delle monete.
Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

1. un'analisi della realtà di riferimento, giungendo alla definizione di uno schema concettuale della base di dati che, a suo motivato giudizio, sia idoneo a gestire la realtà presentata;
2. il relativo schema logico;
3. la definizione in linguaggio SQL di un sottoinsieme delle relazioni della base di dati in cui siano presenti alcune di quelle che contengono vincoli di integrità referenziale e/o vincoli di dominio, se esistenti;
4. le interrogazioni espresse in linguaggio SQL che restituiscono:
   a) l'elenco in ordine alfabetico dei giochi classificati per uno specifico argomento;
   b) la classifica degli studenti di una certa classe virtuale, in base alle monete raccolte per un certo gioco;
   c) il numero di classi in cui è utilizzato ciascun videogioco del catalogo;
5. il progetto di massima della struttura dell'applicazione web per la gestione della realtà sopra presentata;
6. una parte significativa dell'applicazione web che consente l'interazione con la base di dati, utilizzando appropriati linguaggi a scelta sia lato client che lato server.

SECONDA PARTE

I. In relazione al tema proposto nella prima parte, si sviluppi, in un linguaggio a scelta, una porzione di codice significativa delle pagine web necessarie a presentare la classifica generale degli studenti di una certa classe virtuale, in base alle monete raccolte in tutti i videogiochi di quella classe.

II. In relazione al tema proposto nella prima parte, si descriva in che modo è possibile integrare la base di dati sopra sviluppata, per gestire anche i feedback da parte degli studenti sui videogiochi. Ogni feedback è costituito da un punteggio che può andare da 1 a 5 e una descrizione di massimo 160 caratteri. Si descriva anche la struttura delle pagine web dedicate a tale funzionalità, scrivendo in un linguaggio a scelta una porzione di codice significativa di tali pagine.

III. Si descriva, anche attraverso esempi, il concetto di “raggruppamento” nelle interrogazioni SQL, indicando in tale contesto come operano le funzioni di aggregazione e la clausola HAVING.

IV. Data la seguente tabella “Progetti”, il candidato verifichi se soddisfa le proprietà di normalizzazione e proponga uno schema relazionale equivalente che rispetti la terza Forma Normale, motivando le scelte effettuate. Si implementi in linguaggio SQL lo schema relazionale ottenuto.

ID	Titolo	Budget	Tipo	DataInizio	DataFine	Tutor	TelTutor
1	Pensiero computazionale	40.000	PON	20/02/2023	Null	Rossi Mario	345678910
2	Robotica educativa	13.000	PCTO	10/11/2022	30/03/2023	Bianchi Carlo	333444555
3	Tinkering	25.000	PCTO	14/10/2022	20/02/2023	Bianchi Carlo	333444555
4	Realtà virtuale	30.000	PCTO	16/02/2023	30/05/2023	Rossi Mario	345678910

PRIMA PARTE

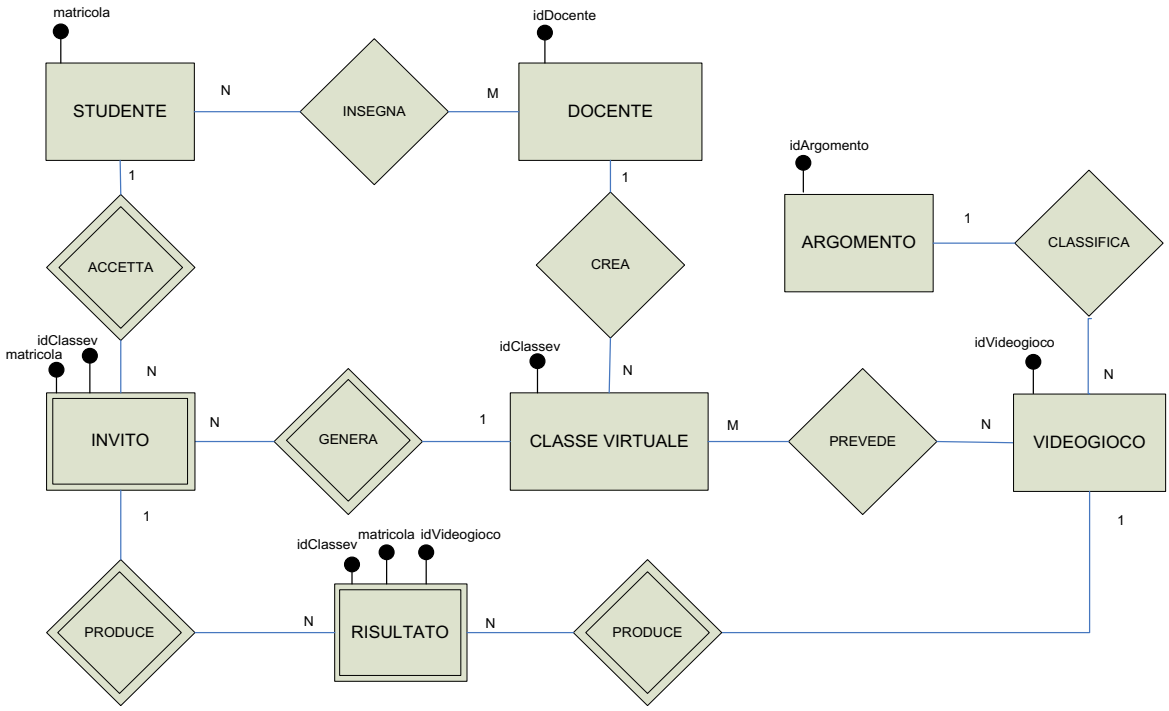
1. Analisi e ipotesi

Per il contesto in cui l'applicazione software da realizzare dovrà essere impiegata è necessario che sia un'applicazione web fruibile da docenti e studenti tramite un normale browser in modalità SAAS (*Software As A Service*): tale applicazione necessita di un database centralizzato, realizzabile ad esempio con DBMS MySQL o Maria-DB che prevedono licenze gratuite, e di un web server con supporto per l'esecuzione di applicazioni lato server per la generazione di pagine web dinamiche (HTML/CSS); è possibile ricorrere ad un web server Apache distribuito con licenza gratuita che integri il modulo per l'esecuzione di script in linguaggio PHP lato server. L'interazione dell'utente con l'applicazione può essere gestita mediante script eseguibili dal browser ed incorporati nelle pagine web generate lato server, utilizzando ad esempio il linguaggio JavaScript.

Nello sviluppare la soluzione che segue per la traccia proposta si è ipotizzato che:

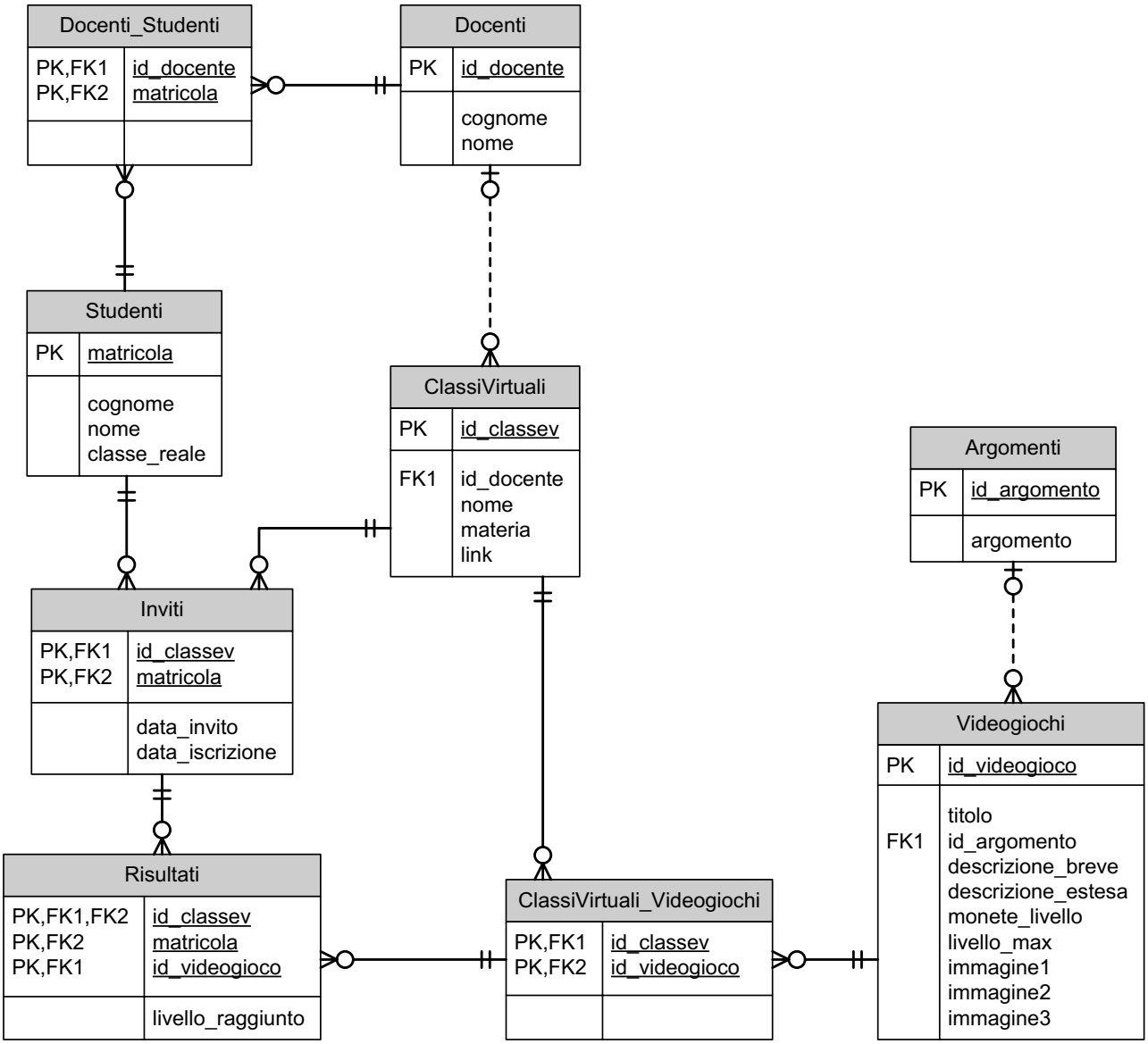
- ogni insegnante possa interagire solo e soltanto con i propri studenti il che significa che l'applicazione software che verrà realizzata proporrà ad ogni docente solo gli studenti delle classi in cui effettivamente insegna (questo vincolo non sarà quindi imposto dalle chiavi delle tabelle);
- una classe virtuale preveda un certo numero di videogiochi ognuno dei quali può essere utilizzato anche in più classi virtuali;
- uno studente possa essere invitato ad iscriversi a una o più classi virtuali;
- ogni invito di iscrizione accettato da parte di uno studente dia luogo ad un risultato per ogni videogioco in cui tale studente si cimenta (utilizzando un campo in cui tenere aggiornato il livello raggiunto dallo studente);
- ogni videogioco preveda un certo numero di livelli ognuno dei quali fornisca allo studente un certo numero di monete virtuali (sempre lo stesso nell'ambito del videogioco indipendentemente dal livello: finché non si supera un livello non si può passare a quello successivo);
- ogni volta che uno studente esegue un gioco l'applicazione software che verrà realizzata aggiorna nella tabella dei risultati riportati dallo studente il livello raggiunto per quel videogioco
- relativamente alla possibilità di utilizzare un QR-code per accedere a una classe virtuale le informazioni contenute in esso costituiscano l'URL necessario per raggiungere la pagina web della classe virtuale: per questa ragione si provvede a memorizzare nel database solo la stringa del link demandando la generazione della relativa immagine del QR-code all'applicazione software
- per brevità nello sviluppo del progetto del database sono state ignorate le informazioni necessarie all'identificazione degli utenti (docenti e studenti) da parte dell'applicazione software (ad esempio: username, password, …)

2. Schema concettuale e logico



La figura mostra lo schema concettuale relativo alla situazione descritta nel testo della prova. Per semplificare sono stati riportati solo gli elementi identificativi propri delle varie entità. In particolare, le entità `INVITO` e `RISULTATO` sono entità deboli perché per essere correttamente identificate hanno bisogno, di combinare le chiavi di `STUDENTE` e `CLASSEVIRTUALE` a cui si aggiunge la chiave di `VIDEOGIOCO` per determinare i risultati.

Nella figura che segue è rappresentato lo schema logico di riferimento per la base di dati da progettare. Le relazioni M:N che sussistono tra docenti e studenti e tra videogiochi e classi virtuali vengono articolate rispettivamente in associazioni 1:N con l'inserimento delle tabelle `Docenti_Studenti` e `ClassiVirtuali_Videogiochi`.
Per quanto riguarda le immagini di ogni videogioco, visto che al massimo possono essere tre, sono previsti tre campi di testo in cui inserire il *pathame* dei file delle immagini nel file-system in una specifica *directory* (in alternativa si sarebbero potuti utilizzare del campi di tipo BLOB per memorizzarle direttamente nel database).



DB-schema
In funzione dello schema logico proposto il DB-schema risultante è quello seguente.

```
/* Tabella: Argomenti. Classifica le varie tipologie di argomenti dei
videogiochi */

    CREATE TABLE Argomenti(
     id_argomento SMALLINT, /* Identificativo argomento */
     argomento VARCHAR(50), /* Descrizione argomento */
     CONSTRAINT PK PRIMARY KEY(id_argomento)
    );

/* Tabella: Docenti. Costituisce l'anagrafica dei docenti */

    CREATE TABLE Docenti(
     id_docente VARCHAR(8), /* Identificativo docente */
     cognome VARCHAR(50), /* Cognome */
     nome VARCHAR(50), /* Nome */
     CONSTRAINT PK PRIMARY KEY(id_docente)
    );


/* Tabella: Studenti. Costituisce l'anagrafica degli studenti */

    CREATE TABLE Studenti(
     matricola VARCHAR(6), /* Matricola studente */
     cognome VARCHAR(25), /* Cognome */
     nome VARCHAR(25), /* Nome */
     classe_reale VARCHAR(5), /* Classe reale di appartenenza */
     CONSTRAINT PK PRIMARY KEY(matricola)
    );

/* Tabella: Videogiochi. Costituisce la libreria dei videogiochi */

    CREATE TABLE Videogiochi(
     id_videogioco VARCHAR(7), /* Identificativo videogioco */
     titolo VARCHAR(50), /* Titolo */
     id_argomento SMALLINT, /* Identificativo argomento */
     descrizione_breve VARCHAR(160), /* Descrizione breve */
     descrizione_estesa VARCHAR(2048), /* Descrizione estesa */
     monete_livello SMALLINT, /* Monete corrisposte per ogni livello */
     livello_max SMALLINT, /* Livello massimo raggiungibile */
     immagine1 VARCHAR(255), /* Pathname prima immagine */
     immagine2 VARCHAR(255), /* Pathname seconda immagine */
     immagine3 VARCHAR(255), /* Pathname terza immagine */
     CONSTRAINT PK PRIMARY KEY(id_videogioco),
     CONSTRAINT Argomenti_Videogiochi FOREIGN KEY(id_argomento)
      REFERENCES Argomenti(id_argomento)
      ON DELETE CASCADE ON UPDATE CASCADE
    );

 /* Tabella: ClassiVirtuali. Costituisce l'anagrafica delle classi virtuali */

    CREATE TABLE ClassiVirtuali(
     id_classev VARCHAR(10), /* Identificativo classe virtuale */
     id_docente VARCHAR(6), /* Identificativo docente proprietario */
     nome VARCHAR(25), /* Nome classe */
     materia VARCHAR(25), /* Materia di riferimento */
     link VARCHAR(128), /* Link alla classe */
     CONSTRAINT PK PRIMARY KEY(id_classev),
     CONSTRAINT Docenti_ClassiVirtuali  FOREIGN KEY(id_docente)
```

```
      REFERENCES Docenti(id_docente)
      ON DELETE CASCADE ON UPDATE CASCADE
    );
```

/* Tabella: ClassiVirtuali_Videogiochi. Costituisce l'associazione tra classi virtuali e i videogiochi che ognuna di esse utilizza */

```
    CREATE TABLE ClassiVirtuali_Videogiochi(
     id_classev VARCHAR(10), /* Identificativo classe virtuale */
     id_videogioco VARCHAR(7), /* Identificativo videogioco */
     CONSTRAINT PK PRIMARY KEY (id_classev,id_videogioco),
     CONSTRAINT ClassiVirtuali_Videogiochi FOREIGN KEY(id_classev)
      REFERENCES ClassiVirtuali(id_classev)
      ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT Videogiochi_ClassiVirtuali_Videogiochi
      FOREIGN KEY(id_videogioco)
      REFERENCES Videogiochi(id_videogioco)
      ON DELETE CASCADE ON UPDATE CASCADE
    );
```

/* Tabella: Docenti_Studenti. Definisce l'associazione tra docenti e relativi studenti */

```
    CREATE TABLE Docenti_Studenti(
     id_docente VARCHAR(8), /* Identificativo docente */
     matricola VARCHAR(6), /* Matricola studente */
     CONSTRAINT PK PRIMARY KEY(id_docente,matricola),
     CONSTRAINT Docenti_Docenti_Studenti FOREIGN KEY(id_docente)
      REFERENCES Docenti(id_docente)
      ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT Studenti_Docenti_Studenti FOREIGN KEY(matricola)
      REFERENCES Studenti(matricola)
      ON DELETE CASCADE ON UPDATE CASCADE
    );
```

/* Tabella: Inviti. Memorizza gli inviti verso gli studenti per l'adesione alla classi virtuali */

```
    CREATE TABLE Inviti(
     id_classev VARCHAR(10), /* Identificativo classe virtuale */
     matricola VARCHAR(6), /* Matricola studente */
     data_invito DATETIME, /* Data di invito da parte del docente */
     data_iscrizione DATETIME, /* Data iscrizione da parte dello studente */
     CONSTRAINT PK PRIMARY KEY(id_classev,matricola),
     CONSTRAINT ClassiVirtuali_Inviti FOREIGN KEY(id_classev)
      REFERENCES ClassiVirtuali(id_classev)
      ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT Studenti_Inviti FOREIGN KEY(matricola)
      REFERENCES Studenti(matricola)
      ON DELETE CASCADE ON UPDATE CASCADE
    );
```

/* Tabella: Risultati. Memorizza i livelli raggiunti da ogni studente nei videogiochi a cui partecipa */

```
    CREATE TABLE Risultati(
     id_classev VARCHAR(10), /* Identificativo classe virtuale */
     matricola VARCHAR(6), /* Matricola studente */
     id_videogioco VARCHAR(7), /* Identificativo videogioco */
     livello_raggiunto SMALLINT, /* Livello raggiuto dallo studente */
     CONSTRAINT PK PRIMARY KEY(id_classev,id_videogioco,matricola),
     CONSTRAINT ClassiVirtuali_Videogiochi_Risultati
```

```
  FOREIGN KEY(id_classev,id_videogioco)
  REFERENCES ClassiVirtuali_Videogiochi(id_classev,id_videogioco)
  ON DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT Inviti_Risultati FOREIGN KEY(id_classev,matricola)
  REFERENCES Inviti(id_classev,matricola)
  ON DELETE CASCADE ON UPDATE CASCADE
);
```

4. Query

(a) Elenco in ordine alfabetico dei giochi classificati per uno specifico argomento

```
SELECT id_videogioco, Argomenti.argomento, titolo, descrizione_breve,
descrizione_estesa, monete_livello
FROM Argomenti INNER JOIN Videogiochi
     ON Argomenti.id_argomento = Videogiochi.id_argomento
WHERE Argomenti.argomento=[…]
ORDER BY titolo;
```

(b) Classifica degli studenti di una certa classe virtuale, in base alle monete raccolte per un certo gioco

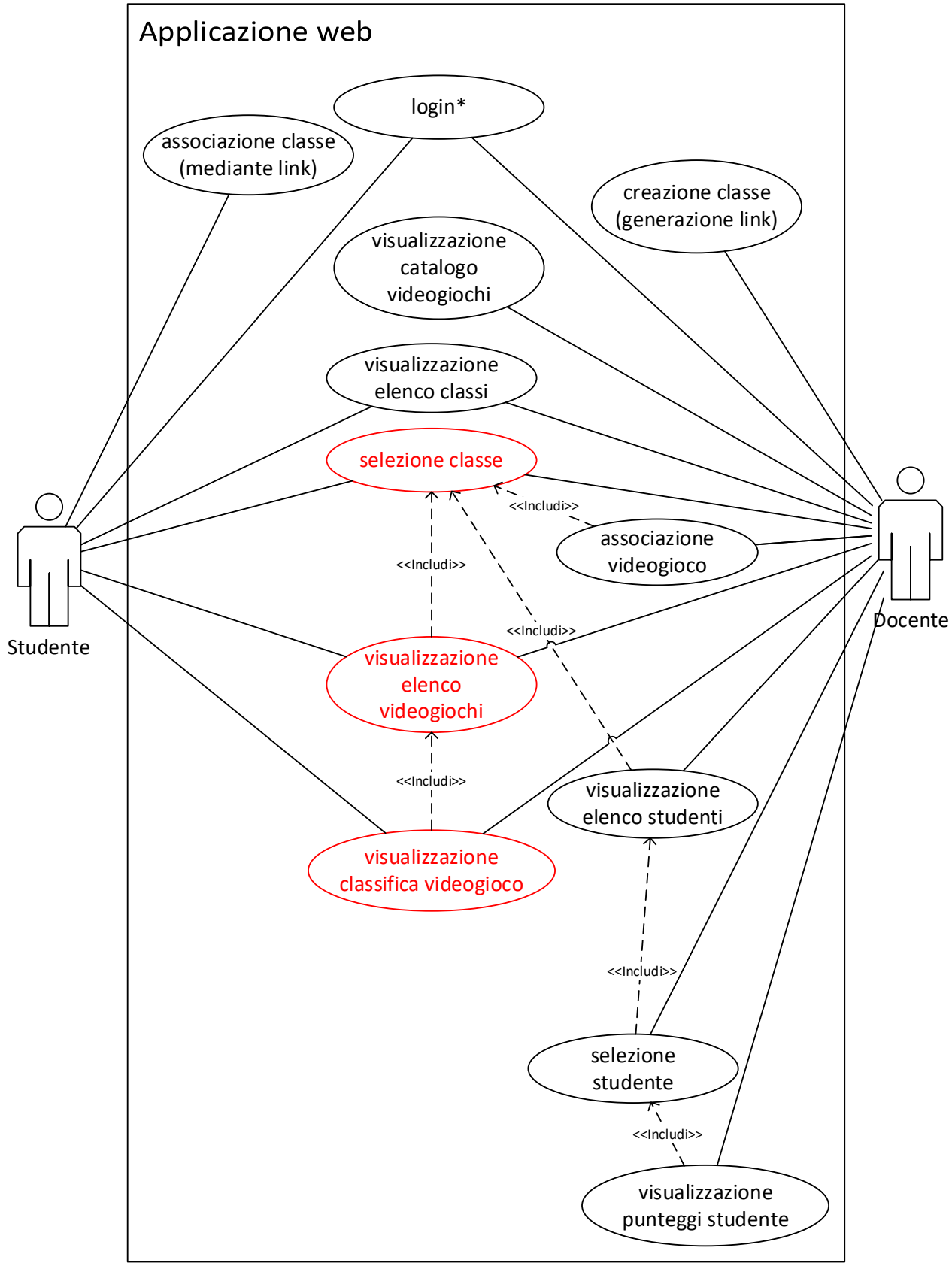
```
SELECT *
FROM (SELECT Risultati.matricola, Studenti.cognome, Studenti.nome,
      SUM(Risultati.livello_raggiunto*Videogiochi.monete_livello)
      AS punteggio_totale
      FROM (Risultati INNER JOIN Videogiochi
            ON Risultati.id_videogioco = Videogiochi.id_videogioco)
            INNER JOIN Studenti ON Risultati.matricola = Studenti.matricola
      WHERE Risultati.id_classev =[…]
        AND Videogiochi.titolo=[…]
      GROUP BY Risultati.matricola, Studenti.cognome, Studenti.nome) AS T
      ORDER BY punteggio_totale DESC;
```

(c) Numero di classi in cui è utilizzato ciascun videogioco del catalogo

```
SELECT Videogiochi.id_videogioco, Videogiochi.titolo, COUNT(*) AS n_classi
FROM Videogiochi INNER JOIN ClassiVirtuali_Videogiochi
     ON Videogiochi.id_videogioco=ClassiVirtuali_Videogiochi.id_videogioco
GROUP BY Videogiochi.id_videogioco, Videogiochi.titolo;
```

5. Progetto di massima della struttura dell'applicazione web per la gestione della realtà descritta.

Il seguente diagramma UML dei casi d'uso rappresenta le interazioni degli utenti (studenti e docenti) con l'applicazione web (sono evidenziati i casi d'uso implementati nel successivo punto 6):



* login è incluso in tutte le azioni rese possibile dall'applicazione

La progettazione delle singole pagine dell'applicazione web sarà effettuata dal un grafico a partire dal diagramma dei casi d'uso illustrato.

6. Codifica applicazione

Come esempio di codifica di una parte significativa dell'applicazione web in linguaggio PHP che interagisce con la base di dati sono presentate le pagine web che supportano l'esecuzione della query (b) del punto 4.

Sono previsti due script di cui il primo permette la selezione della classe virtuale e dei videogiochi attivati per essa (uno o tutti quanti), mentre il secondo presenta la classifica a livello di studenti.

Nel primo script è stata utilizzata una funzione Javascript che, al variare della selezione della classe, valorizza il combo box con l'elenco dei videogiochi attivati per essa.

```
<html>
 <head>
  <title>Selezione classe virtuale-videogioco</title>
 </head>
 <body>
  <?php
   $connection = new mysqli("localhost","root","","edu_games");
   if ($connection->connect_errno)
     die("Errore di connessione al DBMS My-SQL.");
  ?>
  <form action="classificaClGc.php" method="GET" id="formDati">
    <?php

    $query = "SELECT DISTINCT Videogiochi.id_videogioco, videogiochi.titolo,
                              Classivirtuali.id_classev, Classivirtuali.nome
               FROM Videogiochi,Classivirtuali_Videogiochi,Classivirtuali
               WHERE
           Videogiochi.id_videogioco = Classivirtuali_Videogiochi.id_videogioco
               AND
           Classivirtuali.id_classev = Classivirtuali_Videogiochi.id_classev;";
    $result = $connection->query($query);
    if (!$result)
       die("Errore esecuzione query SQL.");
    if ($result->num_rows == 0)
       die("Nessuna associazione classivirtuali-videogiochi censita.");
    $giochi=array();
    $i=0;
    while ($row = $result->fetch_array()){
            $rgioc=array($row[0],$row[1],$row[2],$row[3]);
            $giochi[$i]=$rgioc;
            $i++;
    }
  ?>
  <br>
  <br>
  Selezionare classe virtuale:
  <select name="classe" id="classe" onclick="cambiaGiochi()">
   <?php
    $query = "SELECT DISTINCT id_classev, nome FROM Classivirtuali;";
    $result = $connection->query($query);
    if (!$result)
      die("Errore esecuzione query SQL.");
    if ($result->num_rows == 0)
      die("Nessuna classe virtuale censita.");
    while ($row = $result->fetch_array())
         echo "<option value=\"$row[0]\">$row[1]</option>";
    $result->free();
    $connection->close();
   ?>
  </select>
      Selezionare videogioco (* = Tutti):
  <select name="giochi" id="giochi";>
```

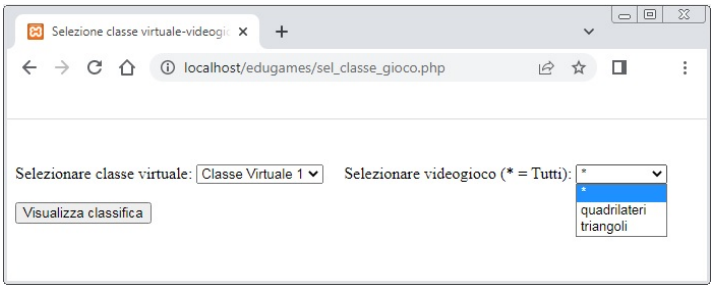
```
    <option value=*>*</option>
   </select>
   <script>
     var elencoGiochi = [];
     var elencoGiochi =  <?php echo json_encode($giochi); ?>;
     function cambiaGiochi(){
        var tc=document.getElementById('classe').value;
        var tg=document.getElementById('giochi');
        while (tg.length) {
             tg.remove(0);
        }
        var opt = new Option('*','*');
        tg.options.add(opt,0);
        l=elencoGiochi.length;
        var i=0;
        while (i<=l){
             if (elencoGiochi[i][2] == tc){
               opt = new Option(elencoGiochi[i][1], elencoGiochi[i][0]);
               tg.options.add(opt,tg.options.length);
        }
        i++
      }
    }
   </script>
   <br>
   <br>
   <input type="submit" value="Visualizza classifica" onclick="chkDati();" />
   <br>
  </form>
 </body>
</html>
```

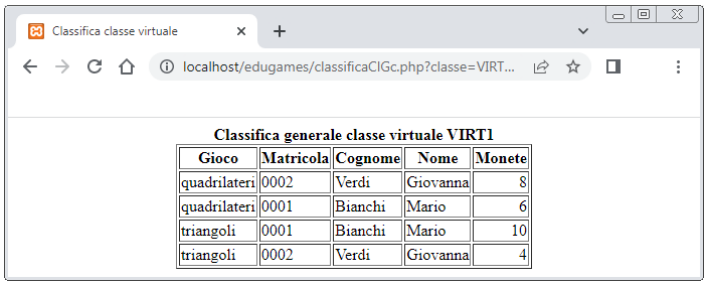
Il secondo script genera la classifica di classe per uno specifico videogioco, o eventualmente tutti, utilizzando, a seconda dei casi, una query opportunamente formulata:

```
<html>
 <head>
  <title>
     Classifica classe virtuale
  </title>
 </head>
 <body>
 <center>
  <?php
   $connection = new mysqli("localhost","root","","edu_games");
   if ($connection->connect_errno)
     die("Errore di connessione al DBMS My-SQL.");
   $classev = $_GET['classe'];
   $gioco = $_GET['giochi'];
   if ($gioco == '*')
     $query = "SELECT *
               FROM (SELECT Risultati.matricola, Studenti.cognome, Studenti.nome,
                     Videogiochi.titolo,
                     SUM(Risultati.livello_raggiunto*Videogiochi.monete_livello)
                     AS punteggio_totale
                     FROM (Risultati INNER JOIN Videogiochi
                     ON Risultati.id_videogioco = Videogiochi.id_videogioco)
                     INNER JOIN Studenti ON Risultati.matricola =
                     Studenti.matricola
                     WHERE Risultati.id_classev ='".$classev."'
                     GROUP BY Videogiochi.titolo,Risultati.matricola,
                     Studenti.cognome, Studenti.nome) AS T
```

```
                    ORDER BY titolo ASC,punteggio_totale DESC;";
   else
       $query = "SELECT *
                 FROM (SELECT Risultati.matricola, Studenti.cognome,
                 Studenti.nome, Videogiochi.titolo,
                 SUM(Risultati.livello_raggiunto*Videogiochi.monete_livello)
                 AS punteggio_totale
                 FROM (Risultati INNER JOIN Videogiochi
                 ON Risultati.id_videogioco = Videogiochi.id_videogioco)
                 INNER JOIN Studenti ON Risultati.matricola = Studenti.matricola
                 WHERE Risultati.id_classev ='".$classev."'
                 AND Risultati.id_videogioco='".$gioco."'
                 GROUP BY Risultati.matricola, Studenti.cognome, Studenti.nome)
                 AS T
                 ORDER BY punteggio_totale DESC;";
   $result = $connection->query($query);
   if (!$result)
     die("Errore esecuzione query SQL.");
   if ($result->num_rows == 0)
     die("Nessun dato trovato.");
  ?>
  <table border>
   <caption><b>
     <?php
       echo "Classifica generale classe virtuale ".$classev; ?>
   </b></caption>
   <thead>
    <tr>
      <th>Gioco</th>
      <th>Matricola</th>
      <th>Cognome</th>
      <th>Nome</th>
      <th>Monete</th>
    </tr>
   </thead>
   <tbody>
    <?php
     while($row = $result->fetch_array()){
    ?>
      <tr>
       <td><?php echo $row['titolo']; ?></td>
       <td><?php echo $row['matricola']; ?></td>
       <td><?php echo $row['cognome']; ?></td>
       <td><?php echo $row['nome']; ?></td>
       <td align="right"><?php echo ($row['punteggio_totale']); ?>
       </td>
      </tr>
    <?php
     }
     echo "</tbody>\n";
     echo "</table>\n";
     $result->free();
     $connection->close();
    ?>
   </center>
 </body>
</html>
```

Le due immagini che seguono rappresentano le pagine web realizzate rispettivamente dai due script PHP:





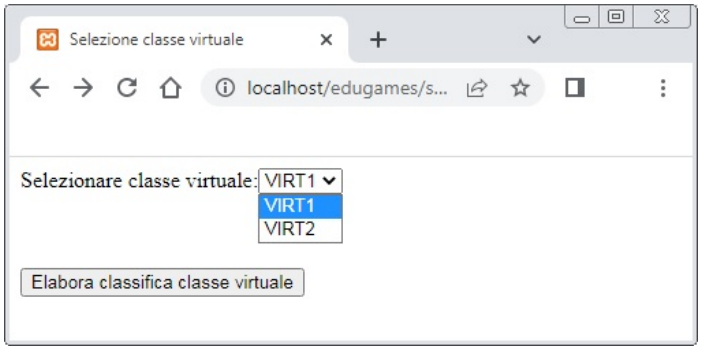
SECONDA PARTE

Punto I

In merito a questo punto sono stati realizzati due script PHP di cui il primo permette all'utente di selezionare la classe virtuale di cui interessa la classifica generale degli studenti ed il secondo che visualizza la classifica richiesta. Entrambi gli script PHP implementano un'interfaccia utente con pagine web completamente prive di struttura grafica.

```
<html>
 <head>
  <title>Selezione classe virtuale</title>
 </head>
 <body>
  <?php
   $connection = new mysqli("localhost","root","","edu_games");
   if ($connection->connect_errno)
     die("Errore di connessione al DBMS My-SQL.");
  ?>
  <form action="classifica.php" method="POST">

  <?php
     //Selezione classe
     echo "Selezionare classe virtuale:";
     echo '<select name="classev"><br>';

     $query = "SELECT DISTINCT id_classev FROM Classivirtuali;";

     $result = $connection->query($query);
     if (!$result)
       die("Errore esecuzione query SQL.");
     if ($result->num_rows == 0)
       die("Nessuna classe virtuale censita.");

     while ($row = $result->fetch_array())
          echo "<option value=\"$row[0]\">$row[0]</option>";
     echo "</select>";
     echo "<br><br>";
     $result->free();
     $connection->close();
    ?>
   <br>
   <br>
   <input type="submit" value="Elabora classifica classe virtuale">
   <br>
  </form>
 </body>
</html>
```

Questo script genera una pagina web dinamica come la seguente:

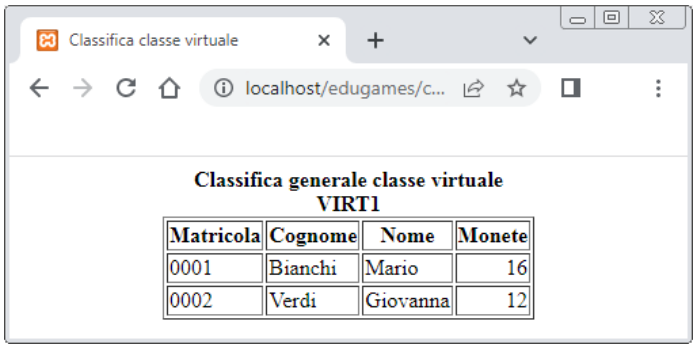


Lo script contenuto nel file `classifica.php`, riportato di seguito, acquisisce tramite il metodo POST il codice della classe selezionata dall'utente e visualizza in forma tabellare la classifica degli studenti.

```
<html>
 <head>
  <title>
     Classifica classe virtuale
  </title>
 </head>
 <body>
 <center>
  <?php
   $connection = new mysqli("localhost","root","","edu_games");
   if ($connection->connect_errno)
     die("Errore di connessione al DBMS My-SQL.");
   $classev = $_POST['classev'];
   $query = "SELECT *
             FROM (SELECT Risultati.matricola, Studenti.cognome, Studenti.nome,
                    SUM(Risultati.livello_raggiunto*Videogiochi.monete_livello)
             AS punteggio_totale
             FROM (Risultati INNER JOIN Videogiochi
                   ON Risultati.id_videogioco = Videogiochi.id_videogioco)
                   INNER JOIN Studenti ON Risultati.matricola =
                                          Studenti.matricola
                   WHERE Risultati.id_classev ='".$classev."'
                   GROUP BY Risultati.matricola, Studenti.cognome, Studenti.nome)
                   AS T
                   ORDER BY punteggio_totale DESC;";

   $result = $connection->query($query);
   if (!$result)
     die("Errore esecuzione query SQL.");
   if ($result->num_rows == 0)
     die("Nessun dato trovato.");
  ?>
  <table border>
   <caption><b>
     <?php
       echo "Classifica generale classe virtuale ".$classev;
     ?>
   </b></caption>
   <thead>
    <tr>
     <th>Matricola</th>
     <th>Cognome</th>
     <th>Nome</th>
     <th>Monete</th>
```

```
    </tr>
   </thead>
   <tbody>
    <?php
     while ($row = $result->fetch_array()){
            ?>
          <tr>
           <td><?php echo $row['matricola']; ?></td>
           <td><?php echo $row['cognome']; ?></td>
           <td><?php echo $row['nome']; ?></td>
           <td align="right"><?php echo ($row['punteggio_totale']);?></td>
          </tr>
    <?php
         }
        echo "</tbody>\n";
        echo "</table>\n";
        $result->free();
        $connection->close();
    ?>
   </center>
 </body>
</html>
```

Questo script genera una pagina web dinamica come la seguente:



Punto II

In funzione della struttura della base di dati sviluppata nella prima parte, per supportare questo punto si può pensare di introdurre tra `Studenti` e `Videogiochi` una tabella `Feedback`. Questa, oltre ai due campi richiesti `punteggio` e `descrizione`, deve prevedere come chiave primaria, la combinazione degli attributi `matricola` e `id_videogioco` così da stabilire tra essa e le due tabelle precedenti due associazioni 1:N (uno studente potrà fornire al più un unico feedback per un certo videogioco). È inoltre ragionevole supporre che gli studenti possano rilasciare feedback solo per quei videogiochi che hanno usato almeno una volta.

In alternativa alla soluzione precedente, ipotizzando che uno studente utilizzi un certo videogioco nell'ambito di una sola classe virtuale, si possono inserire i feedback direttamente nella tabella `Risultati` estendendone la struttura con i campi `punteggio` e `descrizione`. In questo caso, si può prevedere uno script che permette allo studente di selezionare un videogioco già utilizzato recuperandone il codice dalla tabella `Risultati` e compilarne i dati di feedback. Un secondo script, invocato dall'inoltro del primo, riceverà i dati inseriti e provvederà a aggiornare la tabella `Risultati`.

In merito a quest'ultima soluzione, il primo script potrebbe essere implementato come segue:

```
<html>
 <head>
```

```
  <title>Feedback videogiochi</title>
 </head>
 <body>
  <?php
   $connection = new mysqli("localhost","root","","edu_games");
   if ($connection->connect_errno)
     die("Errore di connessione al DBMS My-SQL.");
  ?>
  <form action="up_feedback.php" method="POST">
  <?php
   $matricola = $_POST['matricola'];
   //Selezione videogioco
   $query="SELECT * FROM Risultati,Videogiochi
           WHERE Risultati.id_videogioco=Videogiochi.id_videogioco
                AND matricola='".$matricola."';";
   $result = $connection->query($query);
   if (!$result)
     die("Errore esecuzione query SQL.");
   if ($result->num_rows == 0)
     die("Nessun videogioco disponibile per feedback.");
   echo '<input type="hidden" name="matricola" value="'.$matricola.'">';
   echo "Selezionare videogioco:";
   echo '<select name="videogioco"><br>';
   while ($row = $result->fetch_array())
        echo "<option value=\"$row[2]\">$row[5]</option>";
   echo "</select>";
   echo "<br> <br>Punteggio:   ";
   echo '<input type="number" name="punteggio" min="1" max="5">';
   echo "<br> <br>Descrizione: ";
   echo '<input type="text" name="descrizione" maxlength="160">';
   $result->free();
   $connection->close();
  ?>
  <br><br>
  <input type="submit" value="Lascia feedback">
  <br>
  </form>
 </body>
</html>
```

Lo script `up_feedback.php` che aggiorna effettivamente il database può essere così implementato:

```
<?php
      $matricola = $_POST['matricola'];
      $videogioco = $_ POST['videogioco'];
      $punteggio = $_ POST['punteggio'];
      $descrizione = $_ POST['descrizione'];
      $connection = new mysqli("localhost","root","","edu_games");
      if ($connection->connect_errno)
        die("Errore di connessione al DBMS My-SQL.");
      $query = "UPDATE Risultati
                SET punteggio=".$punteggio.", descrizione==".$descrizione."
                WHERE id_videogioco='".$videogioco."'
                AND matricola='".$matricola."';";
      $connection->query($query);
      $connection->close();
?>
```

Punto III

Il linguaggio SQL fornisce alcune funzioni di aggregazione che possono essere usate nel comando SELECT:

- **SUM**: calcola la somma di un insieme di valori numerici;
- **AVG**: calcola la media aritmetica di un insieme di valori numerici;
- **COUNT**: calcola la cardinalità di un insieme di righe;
- **MAX**: calcola il massimo di un insieme di valori;
- **MIN**: calcola il minimo di un insieme di valori.

Il linguaggio SQL permette di partizionare in gruppi le righe di una tabella utilizzando la clausola **GROUP BY**. A tali gruppi può essere applicata una qualsiasi delle funzioni di aggregazione viste in precedenza.
Ad esempio con la seguente istruzione:

```
SELECT classe, sezione, COUNT(*) AS numero_studenti
FROM Studenti
GROUP BY classe, sezione;
```

è possibile ottenere il numero di alunni per classe (classe + sezione) di una certa scuola.

Una tabella può essere partizionata in gruppi a cui applicare una condizione per selezionarne solo alcuni. Questa selezione viene realizzata impiegando la clausola **HAVING**.
La clausola **HAVING** può essere utilizzata insieme a una qualsiasi funzione di aggregazione, ma sempre associata a una clausola **GROUP BY**.

Ad esempio con la seguente query

```
SELECT classe, sezione
FROM Studenti
GROUP BY classe, sezione
HAVING COUNT(*) >= 25;
```

è possibile individuare le classi di una scuola che abbiano più di 24 studenti.
Si deve fare attenzione alla differenza tra il modo di operare delle due clausole **WHERE** e **HAVING**.
La prima permette la formulazione di condizioni sulla base dei valori dei singoli campi delle diverse righe di una tabella, mentre le condizioni formulabili con la seconda riguardano interi gruppi di righe formati dalla clausola **GROUP BY**.
Per esempio la condizione **HAVING COUNT(*)** utilizzata in precedenza esprime una condizione sulla cardinalità dei gruppi (numero di righe) costruiti dalla clausola **GROUP BY**.
Ovviamente i diversi tipi di condizioni possono agire in sinergia all'interno di una singola query: la clausola **WHERE** seleziona un insieme di righe, la clausola **GROUP BY** provvede al loro partizionamento in gruppi, dei quali solo quelli che soddisfano il criterio specificato dalla condizione **HAVING** saranno quelli selezionati.

Punto IV

Progetti

ID	Titolo	Budget	Tipo	DataInizio	DataFine	Tutor	TelTutor
1	Pensiero computazionale	40.000	PON	20/02/2023	Null	Rossi Mario	345678910
2	Robotica educativa	13.000	PCTO	10/11/2022	30/03/2023	Bianchi Carlo	333444555
3	Tinkering	25.000	PCTO	14/10/2022	20/02/2023	Bianchi Carlo	333444555
4	Realtà virtuale	30.000	PCTO	16/02/2023	30/05/2023	Rossi Mario	345678910

Perché una tabella sia in 1NF normale è necessario:

- individuare una chiave primaria: in questo caso la PK potrebbe essere rappresentata dal campo ID
- tutti gli attributi siano atomici (non ulteriormente scomponibili). In questo caso si osserva che per il nominativo del tutor si può procedere (anche se la cosa, in questo caso, è relativamente significativa) a separare il cognome dal nome.

```
Progetti(id,titolo,budget,tipo,datainizio,datafine,cogntutor,nomtutor, teltutor)
```

Progetti

ID	Titolo	Budget	Tipo	DataInizio	DataFine	CognTutor	NomTutor	TelTutor
1	Pensiero computazionale	40.000	PON	20/02/2023	Null	Rossi	Mario	345678910
2	Robotica educativa	13.000	PCTO	10/11/2022	30/03/2023	Bianchi	Carlo	333444555
3	Tinkering	25.000	PCTO	14/10/2022	20/02/2023	Bianchi	Carlo	333444555
4	Realtà virtuale	30.000	PCTO	16/02/2023	30/05/2023	Rossi	Mario	345678910

Per la 2NF è necessario individuare le dipendenze funzionali complete (un numero di telefono è univoco):

```
id → titolo, budget, tipo, datainizio, datafine, teltutor
teltutor → (cogntutor,nomtutor)
```

per cui si potrebbe proporre la seguente scomposizione:

```
Progetti1(id,titolo,budget,tipo,datainizio,datafine,teltutor)
Tutor(teltutor,cogntutor,nomtutor)
```

Progetti1

ID	Titolo	Budget	Tipo	DataInizio	DataFine	TelTutor
1	Pensiero computazionale	40.000	PON	20/02/2023	Null	345678910
2	Robotica educativa	13.000	PCTO	10/11/2022	30/03/2023	333444555
3	Tinkering	25.000	PCTO	14/10/2022	20/02/2023	333444555
4	Realtà virtuale	30.000	PCTO	16/02/2023	30/05/2023	345678910

Tutor

TelTutor	CognTutor	NomTutor
345678910	Rossi	Mario
333444555	Bianchi	Carlo

Per la 3NF è necessario individuare le dipendenze funzionali transitive. In questo caso non ve ne sono per cui le due tabelle risultano anche in 3NF.

La situazione descritta può essere implementata col DDL di SQL come segue:

```
CREATE TABLE Tutor(
       teltutor VARCHAR(10),
       cogntutor VARCHAR(25),
       nomtutor VARCHAR(25),
       CONSTRAINT PK PRIMARY KEY(teltutor),
      );

CREATE TABLE Progetti1(
       id SMALLINT,
       titolo VARCHAR(50),
       budget DOUBLE,
       tipo VARCHAR(4),
       datainizio DATE,
       datafine DATE,
       teltutor VARCHAR(10),
       CONSTRAINT PK PRIMARY KEY(id),
       CONSTRAINT Progetti1_Tutor FOREIGN KEY(teltutor)
        REFERENCES Tutor(teltutor)
      );
```