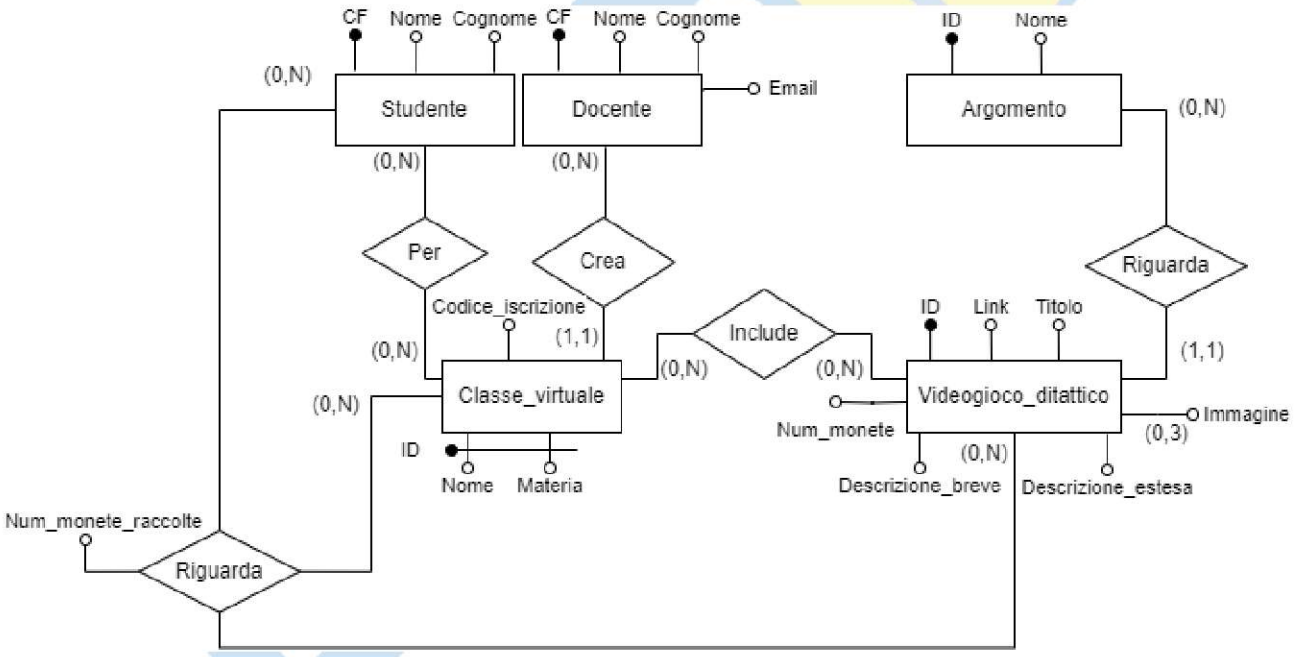
PROPOSTA DI SOLUZIONE PER LA SECONDA PROVA DI MATURITÀ 2023
TRACCIA: Informatica
ARGOMENTO: Piattaforma web per la fruizione di Educational Games

Testo:

PRIMA PARTE

Questa prova richiede la progettazione di una piattaforma web per la fruizione di Educational Games, ovvero videogiochi didattici finalizzati a migliorare l'apprendimento all'interno di un istituto scolastico.

1) Schema concettuale:



2) Schema logico:

Studente(CF, Nome, Cognome, Data_nascita)
Docente(CF, Nome, Cognome, Data_nascita, Email)
Videogioco_ditattico(ID, Link, Titolo, Num_monete, Descrizione_breve, Descrizione_estesa, *ID_argomento,*Immagine1, Immagine2,Immagine3)
Argomento(ID, Nome)

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

Classe_virtuale(Nome, Materia,Codice_iscrizione, *CF_docente*)
Studente-classe(CF_studente, Nome, Materia)
Classe-videogioco(ID_videogioco, Nome, Materia)
Punteggi(CF_studente, Nome, Materia, ID_videogioco, Num_monete_raccolte)

3) SQL:

```
CREATE TABLE Studente (
  CF VARCHAR(16) PRIMARY KEY,
  Nome VARCHAR(50),
  Cognome VARCHAR(50),
  Data_nascita DATE
);

CREATE TABLE Docente (
  CF VARCHAR(16) PRIMARY KEY,
  Nome VARCHAR(50),
  Cognome VARCHAR(50),
  Data_nascita DATE,
  Email VARCHAR(100)
);

CREATE TABLE Videogioco_didattico (
  ID INT PRIMARY KEY,
  Link VARCHAR(200),
  Titolo VARCHAR(100),
  Num_monete INT,
  Descrizione_breve VARCHAR(160),
  Descrizione_estesa TEXT,
  ID_argomento INT,
  Immagine1 VARCHAR(200),
  Immagine2 VARCHAR(200),
  Immagine3 VARCHAR(200),
  FOREIGN KEY (ID_argomento) REFERENCES Argomento(ID)
);

CREATE TABLE Argomento (
  ID INT PRIMARY KEY,
  Nome VARCHAR(100)
);

CREATE TABLE Classe_virtuale (
```

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

```
  Nome VARCHAR(100),
  Materia VARCHAR(100),
  Codice_iscrizione VARCHAR(10),
  CF_docente VARCHAR(16),
  PRIMARY KEY (Nome, Materia),
  FOREIGN KEY (CF_docente) REFERENCES Docente(CF)
);

CREATE TABLE Studente_classe (
  CF_studente VARCHAR(16),
  Nome VARCHAR(100),
  Materia VARCHAR(100),
  PRIMARY KEY (CF_studente, Nome, Materia),
  FOREIGN KEY (CF_studente) REFERENCES Studente(CF),
  FOREIGN KEY (Nome, Materia) REFERENCES Classe_virtuale(Nome, Materia)
);

CREATE TABLE Classe_videogioco (
  ID_videogioco INT,
  Nome VARCHAR(100),
  Materia VARCHAR(100),
  PRIMARY KEY (ID_videogioco, Nome, Materia),
  FOREIGN KEY (ID_videogioco) REFERENCES Videogioco_didattico(ID),
  FOREIGN KEY (Nome, Materia) REFERENCES Classe_virtuale(Nome
 FOREIGN KEY (Nome, Materia) REFERENCES Classe_virtuale(Nome, Materia)
);

CREATE TABLE Punteggi (
  CF_studente VARCHAR(16),
  Nome VARCHAR(100),
  Materia VARCHAR(100),
  ID_videogioco INT,
  Num_monete_raccolte INT,
  PRIMARY KEY (CF_studente, Nome, Materia, ID_videogioco),
  FOREIGN KEY (CF_studente, Nome, Materia) REFERENCES
Studente_classe(CF_studente, Nome, Materia),
  FOREIGN KEY (ID_videogioco) REFERENCES Videogioco_didattico(ID)
);
```

Soluzione a cura di

Daniele Costoli

Insegnante di Informatica su Ripetizioni.it

4) Query SQL:
   a) Elenco giochi in base all'argomento:

```
SELECT Titolo
FROM Videogioco_didattico v
JOIN Argomento a ON v.ID_argomento = a.ID
WHERE a.Nome = 'nome_argomento'
ORDER BY Titolo ASC;
```

   b) Classifica studenti di una classe per uno specifico gioco:

```
SELECT s.CF, s.Nome, s.Cognome, p.Num_monete_raccolte
FROM Studente s
JOIN Punteggi p ON s.CF = p.CF_studente
WHERE p.Materia = 'materia' AND p.Nome = 'nome' AND p.ID_videogioco =
'id_videogioco'
ORDER BY p.Num_monete_raccolte DESC;
```

   c) Trovare il numero di classe in cui è utilizzato ciascun videogioco del catalogo :

```
SELECT Videogioco_didattico.ID, Videogioco_didattico.Titolo,
COUNT(Classe_videogioco.Nome) AS Numero_classi
FROM Videogioco_didattico
LEFT JOIN Classe_videogioco ON Videogioco_didattico.ID =
Classe_videogioco.ID_videogioco
GROUP BY Videogioco_didattico.ID, Videogioco_didattico.Titolo
ORDER BY Videogioco_didattico.Titolo ASC;
```

5) Progetto di massima per la struttura dell'applicazione web per la gestione del servizio
Di seguito è presentato un progetto di massima per la struttura dell'applicazione web per la gestione del servizio di Educational Games:

1. Frontend:
   - Pagina di accesso: permette agli utenti di accedere all'applicazione utilizzando le loro credenziali.
   - Dashboard dell'amministratore: fornisce all'amministratore una panoramica delle funzionalità e dei dati dell'applicazione. Include opzioni per gestire gli utenti, le classi, i docenti, i videogiochi, gli argomenti e le statistiche.
   - Dashboard del docente: consente ai docenti di gestire le proprie classi, i videogiochi assegnati e visualizzare le statistiche degli studenti.
   - Dashboard dello studente: mostra agli studenti le classi a cui sono iscritti, i videogiochi assegnati, i punteggi e le classifiche.

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

- Pagina di ricerca dei videogiochi: consente ai docenti di cercare e selezionare i videogiochi didattici in base agli argomenti desiderati.
- Pagina di dettaglio del videogioco: mostra informazioni dettagliate sul videogioco selezionato, tra cui la descrizione, le immagini e il numero di monete.
- Pagina di gestione delle classi: consente agli amministratori e ai docenti di creare, modificare e eliminare classi virtuali, nonché di invitare gli studenti a iscriversi.
- Pagina di gestione dei punteggi: permette agli amministratori e ai docenti di visualizzare i punteggi degli studenti per ciascun videogioco e di generare le classifiche.

2. Backend:
- Gestione degli utenti: gestisce l'autenticazione e l'autorizzazione degli utenti.
- Gestione delle classi: permette la creazione, la modifica e l'eliminazione delle classi virtuali, nonché l'invito degli studenti tramite codice di iscrizione.
- Gestione dei videogiochi: gestisce l'aggiunta, la modifica e l'eliminazione dei videogiochi didattici, inclusi i dettagli come il titolo, le monete, le descrizioni e le immagini.
- Gestione degli argomenti: consente di gestire l'elenco degli argomenti per classificare i videogiochi didattici.
- Gestione dei punteggi: registra e calcola i punteggi degli studenti per ciascun videogioco, consentendo la generazione delle classifiche.

6) Parte dell'applicativo web per interazione con la base di dati:

Lato client:

```
// Codice JavaScript per gestire l'interazione con il backend
// Esempio di funzione per ottenere la lista dei videogiochi in base a un argomento specifico

// Funzione per ottenere la lista dei videogiochi in base all'argomento
function getVideogiochiByArgomento(argomento) {
  // Effettua una richiesta al backend per ottenere i videogiochi in base all'argomento
  fetch(`/api/videogiochi?argomento=${argomento}`)
    .then(response => response.json())
    .then(data => {
      // Manipola i dati ottenuti dal backend
      // Ad esempio, visualizza la lista dei videogiochi sul frontend
      renderVideogiochi(data);
    })
    .catch(error => {
      // Gestisce eventuali errori durante la richiesta al backend
      console.error('Errore nella richiesta dei videogiochi:', error);
    });
}
```

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

```
// Esempio di utilizzo della funzione getVideogiochiByArgomento
const argomentoDesiderato = 'Matematica';
getVideogiochiByArgomento(argomentoDesiderato);
```

Lato server:

```
// Codice JavaScript per gestire le richieste dal client e interagire con la base di dati
// Esempio per ottenere i videogiochi in base all'argomento

// Importa il modulo per gestire le richieste HTTP
const express = require('express');
const app = express();

// Esempio di endpoint per ottenere i videogiochi in base all'argomento
app.get('/api/videogiochi', (req, res) => {
  // Ottieni l'argomento dalla richiesta del client
  const argomentoDesiderato = req.query.argomento;

  // Effettua una query al database per ottenere i videogiochi in base all'argomento
  // Esempio di query SQL con il modulo per l'interazione con il database
  db.query('SELECT * FROM Videogioco_ditattico WHERE ID_argomento = ?',
[argomentoDesiderato], (error, results) => {
    if (error) {
      // Gestisci eventuali errori nella query al database
      console.error('Errore nella query dei videogiochi:', error);
      res.status(500).json({ error: 'Errore nella query dei videogiochi' });
    } else {
      // Invia i dati ottenuti dal database come risposta al client
      res.json(results);
    }
  });
});

// Avvia il server che ascolta le richieste HTTP
app.listen(3000, () => {
  console.log('Server avviato e in ascolto sulla porta 3000');
});
```

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

SECONDA PARTE

I. Lato client:

```
// Codice JavaScript per gestire l'interazione con il backend
// Esempio di funzione per ottenere la classifica generale degli studenti di una classe virtuale

// Funzione per ottenere la classifica generale degli studenti di una classe virtuale
function getClassificaGenerale(classeVirtuale) {
  // Effettua una richiesta al backend per ottenere la classifica generale degli studenti
  fetch(`/api/classi/${classeVirtuale}/classifica`)
    .then(response => response.json())
    .then(data => {
      // Manipola i dati ottenuti dal backend
      // Ad esempio, visualizza la classifica generale degli studenti sul frontend
      renderClassificaGenerale(data);
    })
    .catch(error => {
      // Gestisce eventuali errori durante la richiesta al backend
      console.error('Errore nella richiesta della classifica generale:', error);
    });
}

// Esempio di utilizzo della funzione getClassificaGenerale per una classe virtuale specifica
const classeVirtualeDesiderata = '3B matematica';
getClassificaGenerale(classeVirtualeDesiderata);
```

Lato server:

```
// Codice JavaScript per gestire le richieste dal client e interagire con la base di dati
// Esempio di endpoint per ottenere la classifica generale degli studenti di una classe virtuale

// Importa il modulo per gestire le richieste HTTP
const express = require('express');
const app = express();

// Esempio di endpoint per ottenere la classifica generale degli studenti di una classe virtuale
app.get('/api/classi/:classeVirtuale/classifica', (req, res) => {
  // Ottieni il nome della classe virtuale dai parametri della richiesta del client
  const classeVirtualeDesiderata = req.params.classeVirtuale;
```

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

```
  // Effettua una query al database per ottenere la classifica generale degli studenti
  // Esempio di query SQL per ottenere la classifica ordinata in base al numero di monete raccolte
  db.query(`
    SELECT Nome, Cognome, Num_monete_raccolte
    FROM Studente
    WHERE NomeClasse = ?
    ORDER BY Num_monete_raccolte DESC
  `, [classeVirtualeDesiderata], (error, results) => {
    if (error) {
      // Gestisci eventuali errori nella query al database
      console.error('Errore nella query della classifica generale:', error);
      res.status(500).json({ error: 'Errore nella query della classifica generale' });
    } else {
      // Invia i dati ottenuti dal database come risposta al client
      res.json(results);
    }
  });
});

// Avvia il server che ascolta le richieste HTTP
app.listen(3000, () => {
  console.log('Server avviato e in ascolto sulla porta 3000');
});
```

III. Il concetto di "raggruppamento" nelle interrogazioni SQL consente di combinare e aggregare i dati in base a una o più colonne. Questo permette di ottenere risultati aggregati e filtrare i risultati in base a determinate condizioni utilizzando la clausola HAVING.

Supponiamo di avere una tabella "Studenti" con le colonne "Nome", "Classe" e "Voto". Vogliamo calcolare la media dei voti per ogni classe e selezionare solo le classi con una media dei voti superiore a 7.

L'interrogazione SQL potrebbe essere la seguente:

```
SELECT Classe, AVG(Voto) AS MediaVoti
FROM Studenti
GROUP BY Classe
HAVING AVG(Voto) > 7;
```

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it

In questo caso, la clausola "GROUP BY Classe" raggruppa i dati in base alla colonna "Classe". La funzione di aggregazione "AVG" viene utilizzata per calcolare la media dei voti per ogni classe. La clausola "HAVING AVG(Voto) > 7" filtra i risultati del raggruppamento, selezionando solo le classi con una media dei voti superiore a 7.

Un altro esempio potrebbe essere il seguente, supponiamo di avere una tabella "Prodotti" con le colonne "NomeProdotto", "Categoria" e "Prezzo". Vogliamo selezionare solo le categorie di prodotti che hanno almeno 5 prodotti con un prezzo superiore a 100.

L'interrogazione SQL potrebbe essere la seguente:

```
SELECT Categoria
FROM Prodotti
GROUP BY Categoria
HAVING COUNT(*) > 5 AND MAX(Prezzo) > 100;
```

In questo caso, la clausola "GROUP BY Categoria" raggruppa i dati in base alla colonna "Categoria". La clausola "HAVING COUNT(*) > 5" filtra i risultati del raggruppamento, selezionando solo le categorie con almeno 5 prodotti. Inoltre, la clausola "HAVING MAX(Prezzo) > 100" filtra ulteriormente i risultati, selezionando solo le categorie con almeno un prodotto con prezzo superiore a 100.

In entrambi gli esempi, la clausola HAVING viene utilizzata per applicare condizioni ai risultati del raggruppamento, consentendo di filtrare i dati aggregati in base alle esigenze specifiche dell'applicazione.

Soluzione a cura di
Daniele Costoli
Insegnante di Informatica su Ripetizioni.it