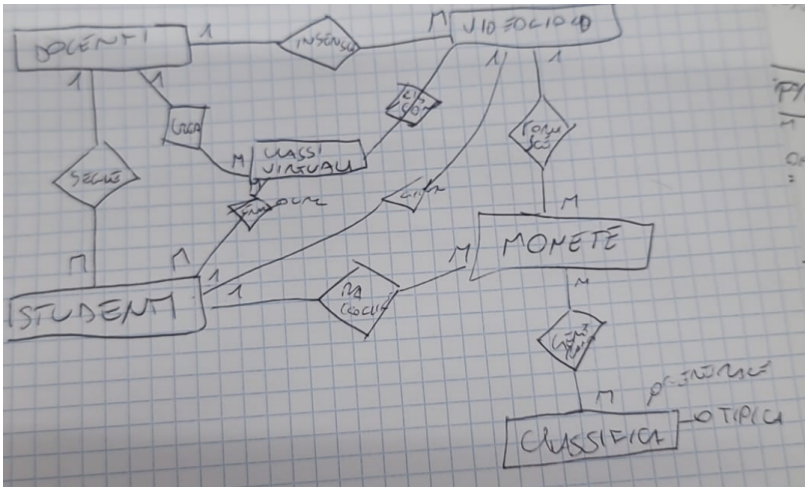
Una scuola vuole progettare una piattaforma web per la fruizione di *educational games* (ovvero videogiochi in ambito educativo), per migliorare l'apprendimento nelle varie materie.

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.

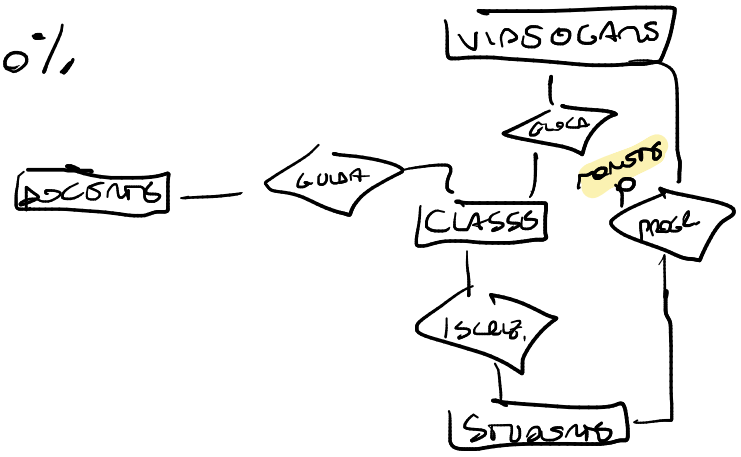
Uno studente si iscriverà sulla piattaforma alle classi cui è stato invitato (es: 3B matematica, 3B italiano ...) tramite il relativo codice iscrizione, e all'interno di ciascuna classe troverà i link ai videogiochi didattici proposti dal docente. Svolgendo ciascun videogioco, lo studente potrà raccogliere 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.



GTA 5 ← 60 / 57 / 56

63/63 = 100%

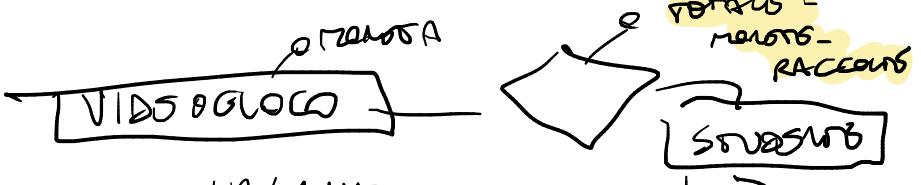


Relazioni

1. **DOCENTE** (1) → (N) **CLASSE_VIRTUALE**
2. **STUDENTE** (N) ← (N) **CLASSE_VIRTUALE** tramite **ISCRIZIONE**
3. **VIDEOGAME** (N) ← (N) **CLASSE_VIRTUALE** tramite **VIDEOGAME_CLASSE**
4. **STUDENTE** (1) → (N) **PROGRESSO_STUDENTE**
5. **VIDEOGAME** (1) → (N) **PROGRESSO_STUDENTE**
6. **CLASSE_VIRTUALE** (1) → (N) **PROGRESSO_STUDENTE**

PROGRESSO = CLASSIFICA

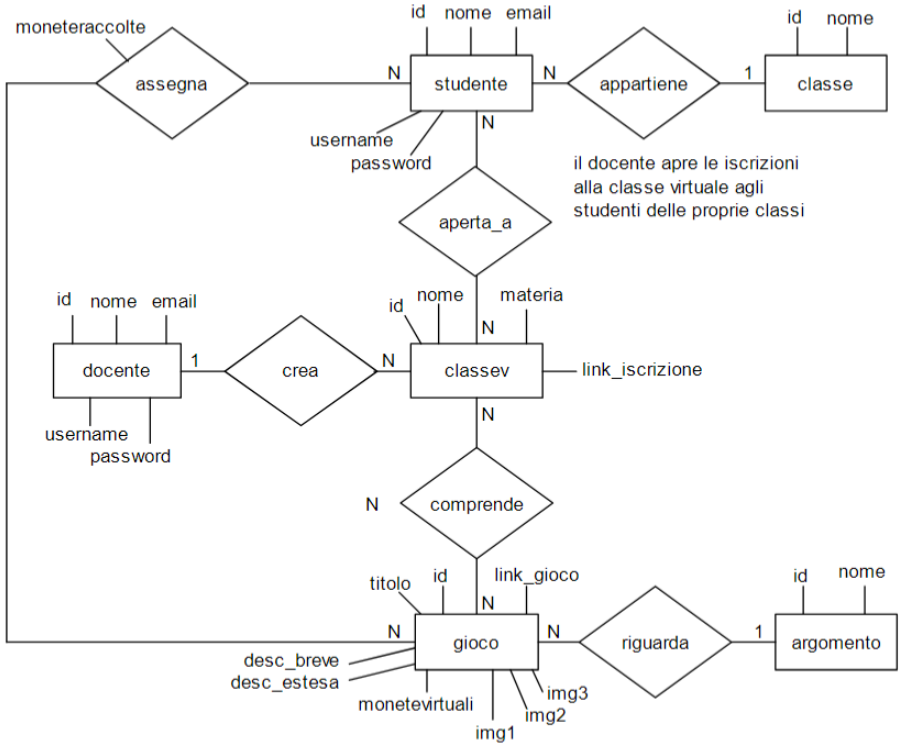
MONETE_RACCOLTE = MOSTRO IL CAMPO "TOTALE" ALL'INTERNO DELLA RELAZIONE



(GTA — HA INIZIATO IL GIOCO!
— 10 MISSIONI COMPLETATE!)

https://toghiri.github.io/pages/appunti.html

APPUNTI V2



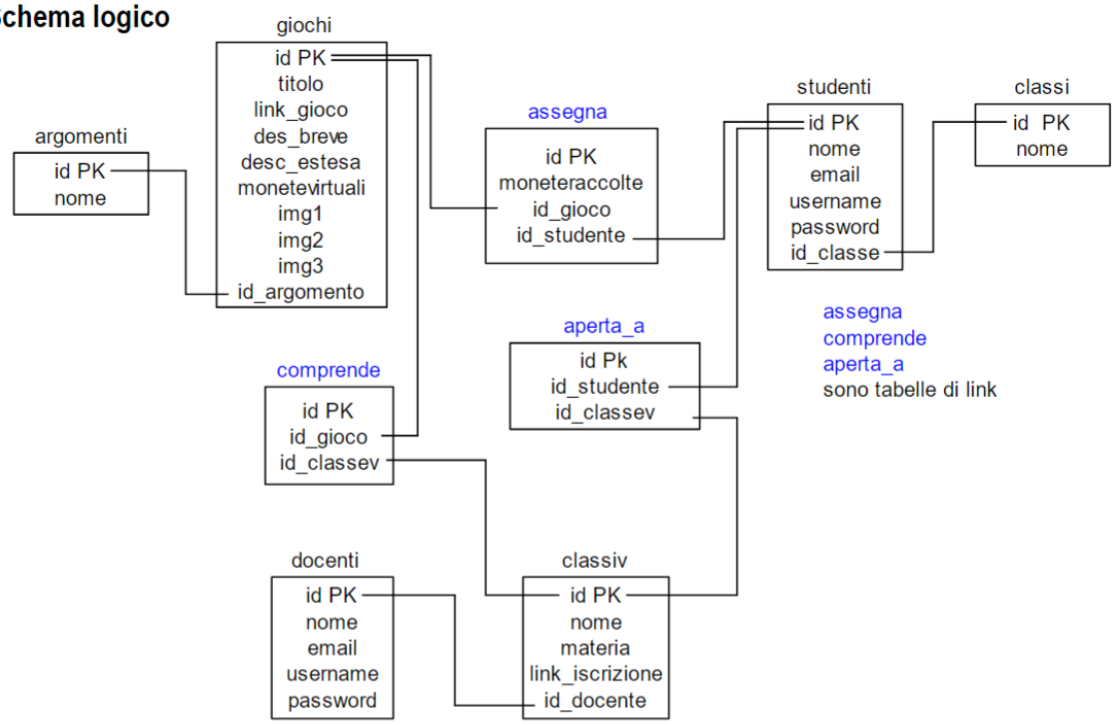

```
-- Tabella DOCENTE
CREATE TABLE DOCENTE (
    id_docente INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    cognome VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    data_registrazione TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

PUNTO = CREAZIONE TABELLE

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;

```
-- Tabella STUDENTE
CREATE TABLE STUDENTE (
    id_studente INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    cognome VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    data_registrazione TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabella VIDEOGAME
CREATE TABLE VIDEOGAME (
    id_videogame INT PRIMARY KEY AUTO_INCREMENT,
    titolo VARCHAR(100) NOT NULL,
    descrizione TEXT,
    argomenti JSON,
    caratteristiche JSON,
    numero_monete_virtuali INT DEFAULT 0,
    data_creazione TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabella CLASSE_VIRTUALE
CREATE TABLE CLASSE_VIRTUALE (
    id_classe INT PRIMARY KEY AUTO_INCREMENT,
    nome_classe VARCHAR(100) NOT NULL,
    materia VARCHAR(50) NOT NULL,
    codice_iscrizione VARCHAR(20) UNIQUE NOT NULL,
    link_iscrizione VARCHAR(255),
    id_docente INT NOT NULL,
    data_creazione TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    attiva BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (id_docente) REFERENCES DOCENTE(id_docente) ON DELETE CASCADE
);
```

Schema logico



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.

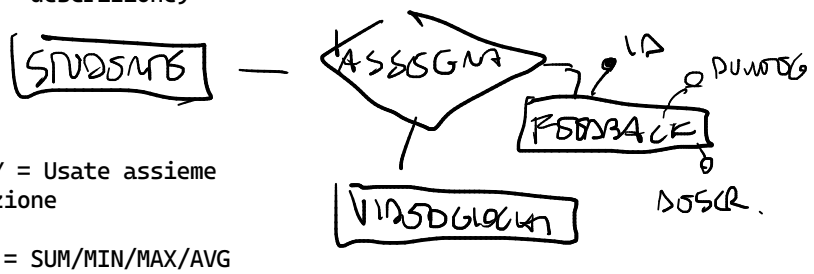
Punto 1: Presenta la classifica
degli studenti
per un certo videogioco

```
<?php

$videogioco = $_GET("videogioco"); // input

(1) Connessione
$connessione = mysqli_connect("db", "localhost", "root", "");

(2) Creazione query ed esecuzione

$query = "SELECT \* FROM Classe_virtuale";

$result = mysqli_query($connessione, $query);

while(mysqli_num_rows() > 0){
        $row = mysqli_fetch_assoc($result, MYSQLI_ASSOC);

        echo "<table> <tr>".$row.codice."</tr>";
}

?>
```

Seconda Parte:

(II) Gestione dei feedback nel DB?

- Relazione chiamata "Feedback" collegata a studenti
e videogiochi (punteggio + descrizione)



(III)

Raggruppamento = GROUP BY = Usate assieme
alle funzioni di aggregazione

Funzioni di aggregazione = SUM/MIN/MAX/AVG
HAVING = Come il WHERE ma per le funzioni di aggregazione =
Condizione sulle funzioni di aggregazione

Caso d'uso di HAVING: Il nome dei giochi con più di 10 giocatori

- Condizione sul conteggio = Usi HAVING
- Se vuoi stampare il conteggio come output (colonna) allora lo piazzi
nel SELECT

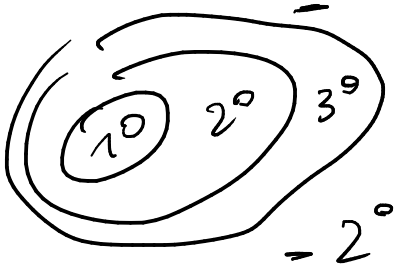
```
SELECT COUNT(*) AS Numero, Nome
FROM Giochi G
JOIN Studenti S ON G.ID = S.Gioco
GROUP BY Nome
HAVING COUNT(*) > 10;
```

NON puoi fare WHERE COUNT, WHERE MAX ⟶ Ecco perché usi Q1

(IV)

Terza Forma Normale ⟶ SQL

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	34567891



- 1° F.N = NO ATTRIBUTI MULTIVALORI

INDIRIZZO → CAP / VIA / CIVICO] SPOSTI IN + ATTRIBUTI

- 2° F.N = NO DIP. PARZIALI TRA TABELLE

- 3 F.N = TUTTO DIPENDE SOLO DALLA CHIAVE