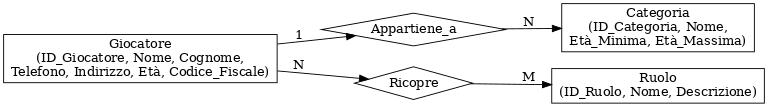
Modello E-R - Esercizio 1 La società calcistica Giorgi Calcio vuole creare un database in grado di gestire le informazioni dei: Giocatori iscritti, memorizzando il nome, il cognome, il telefono, l'indirizzo di residenza, l'età ed il codice fiscale; La categoria calcistica a cui appartiene il giocatore. Della categoria calcistica memorizzare il nome, il numero di anni minimo e massimo che i giocatori possono avere per far parte di tale categoria; Il ruolo ricoperto dal giocatore, memorizzandone il nome e la descrizione del ruolo. Nota Bene: In fase di iscrizione ad un giocatore viene assegnata una sola categoria ed almeno un ruolo che è in grado di ricoprire.

**Relazioni**:

1. **Giocatore-Categoria**:
   * Relazione **"Appartiene\_a"** tra **Giocatore** e **Categoria**.
   * Un giocatore appartiene a una sola categoria (1:1).
   * Una categoria può avere più giocatori (1:N).
2. **Giocatore-Ruolo**:
   * Relazione **"Ricopre"** tra **Giocatore** e **Ruolo**.
   * Un giocatore deve ricoprire almeno un ruolo, ma può ricoprire più ruoli (N:M).
   * Un ruolo può essere ricoperto da più giocatori (N:M).

**Diagramma E-R (Descrizione testuale)**:

* **Giocatore** è connesso a **Categoria** tramite una relazione **"Appartiene\_a"** con molteplicità (1:N).
* **Giocatore** è connesso a **Ruolo** tramite una relazione **"Ricopre"** con molteplicità (N:M).



-- Tabella Giocatore

CREATE TABLE Giocatore (

ID\_Giocatore INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50) NOT NULL,

Cognome VARCHAR(50) NOT NULL,

Telefono VARCHAR(15) NOT NULL,

Indirizzo VARCHAR(100) NOT NULL,

Eta INT NOT NULL,

Codice\_Fiscale VARCHAR(16) UNIQUE NOT NULL

);

-- Tabella Categoria

CREATE TABLE Categoria (

ID\_Categoria INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50) NOT NULL,

Eta\_Minima INT NOT NULL,

Eta\_Massima INT NOT NULL

);

-- Tabella Ruolo

CREATE TABLE Ruolo (

ID\_Ruolo INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50) NOT NULL,

Descrizione TEXT NOT NULL

);

-- Relazione Giocatore-Categoria (1:N)

ALTER TABLE Giocatore

ADD COLUMN ID\_Categoria INT NOT NULL,

ADD CONSTRAINT FK\_Giocatore\_Categoria

FOREIGN KEY (ID\_Categoria) REFERENCES Categoria(ID\_Categoria);

-- Relazione Giocatore-Ruolo (N:M)

CREATE TABLE Giocatore\_Ruolo (

ID\_Giocatore INT NOT NULL,

ID\_Ruolo INT NOT NULL,

PRIMARY KEY (ID\_Giocatore, ID\_Ruolo),

CONSTRAINT FK\_Giocatore\_Ruolo\_Giocatore

FOREIGN KEY (ID\_Giocatore) REFERENCES Giocatore(ID\_Giocatore),

CONSTRAINT FK\_Giocatore\_Ruolo\_Ruolo

FOREIGN KEY (ID\_Ruolo) REFERENCES Ruolo(ID\_Ruolo)

);

**Descrizione dello Schema Logico**

1. **Tabella Giocatore**:
   * Contiene le informazioni di ciascun giocatore, incluso un riferimento a una categoria.
   * La relazione con la tabella Categoria è gestita tramite la colonna ID\_Categoria.
2. **Tabella Categoria**:
   * Memorizza le categorie calcistiche con i rispettivi limiti di età.
3. **Tabella Ruolo**:
   * Contiene i dati relativi ai ruoli che i giocatori possono ricoprire.
4. **Tabella Giocatore\_Ruolo**:
   * Gestisce la relazione molti-a-molti tra Giocatore e Ruolo.

### ****Schema Relazionale****

1. **Giocatore**  
   Giocatore(ID\_Giocatore (PK),Nome,Cognome,Telefono,Indirizzo,Eta,Codice\_Fiscale,ID\_Categoria (FK))\text{Giocatore}(\text{ID\\_Giocatore (PK)}, \text{Nome}, \text{Cognome}, \text{Telefono}, \text{Indirizzo}, \text{Eta}, \text{Codice\\_Fiscale}, \text{ID\\_Categoria (FK)})Giocatore(ID\_Giocatore (PK),Nome,Cognome,Telefono,Indirizzo,Eta,Codice\_Fiscale,ID\_Categoria (FK))
2. **Categoria**  
   Categoria(ID\_Categoria (PK),Nome,Eta\_Minima,Eta\_Massima)\text{Categoria}(\text{ID\\_Categoria (PK)}, \text{Nome}, \text{Eta\\_Minima}, \text{Eta\\_Massima})Categoria(ID\_Categoria (PK),Nome,Eta\_Minima,Eta\_Massima)
3. **Ruolo**  
   Ruolo(ID\_Ruolo (PK),Nome,Descrizione)\text{Ruolo}(\text{ID\\_Ruolo (PK)}, \text{Nome}, \text{Descrizione})Ruolo(ID\_Ruolo (PK),Nome,Descrizione)
4. **Giocatore\_Ruolo**  
   Giocatore\_Ruolo(ID\_Giocatore (FK),ID\_Ruolo (FK),PK(ID\_Giocatore, ID\_Ruolo))

### ****Descrizione dello Schema Relazionale****

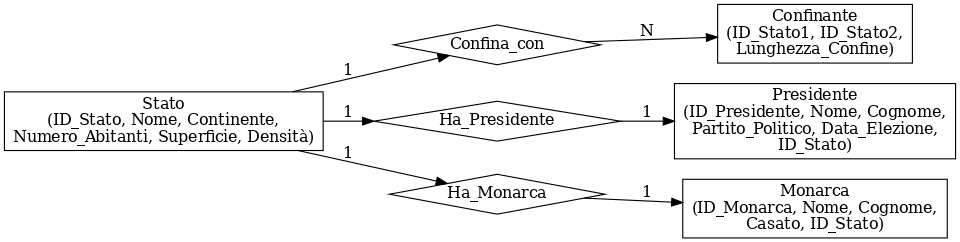
1. **Giocatore**:
   * Memorizza i dati personali di ogni giocatore.
   * La colonna ID\_Categoria è una chiave esterna che collega ogni giocatore alla categoria calcistica di appartenenza.
2. **Categoria**:
   * Contiene informazioni sulle categorie calcistiche, inclusi i limiti di età minimi e massimi.
3. **Ruolo**:
   * Memorizza i dati sui ruoli calcistici e le loro descrizioni.
4. **Giocatore\_Ruolo**:
   * Implementa la relazione molti-a-molti tra giocatori e ruoli.
   * Ogni riga rappresenta un'associazione tra un giocatore e un ruolo.

Si vuole progettare una base di dati per gestire le informazioni relative agli Stati (nome, continente, numero di

abitanti, superficie e densità) , a quelli a essi confinanti e ai rispettivi presidenti, nel caso di repubblica, di cui si

conosce il nome, il cognome e l’eventuale partito politico di appartenenza e la data di elezione, oppure a un

re (o una regina), nel caso di monarchia, del quale si conosce il nome, il cognome e il casato di appartenenza.



## **Modello E-R**

### ****Entità e Attributi****:

1. **Stato**
   * ID\_Stato (Chiave primaria)
   * Nome
   * Continente
   * Numero\_Abitanti
   * Superficie
   * Densità
2. **Confinante**
   * ID\_Stato1 (Chiave primaria composta, riferimento a Stato)
   * ID\_Stato2 (Chiave primaria composta, riferimento a Stato)
   * Lunghezza\_Confine (opzionale)
3. **Presidente**
   * ID\_Presidente (Chiave primaria)
   * Nome
   * Cognome
   * Partito\_Politico (opzionale)
   * Data\_Elezione
   * ID\_Stato (Chiave esterna verso Stato)
4. **Monarca**
   * ID\_Monarca (Chiave primaria)
   * Nome
   * Cognome
   * Casato
   * ID\_Stato (Chiave esterna verso Stato)

### ****Relazioni****:

1. **Stato-Confinante**:
   * Relazione **"Confina\_con"** tra **Stato** e sé stesso (autoreferenziale).
   * Attributo: Lunghezza\_Confine (opzionale).
   * Molteplicità: Uno Stato può confinare con N altri Stati.
2. **Stato-Presidente**:
   * Relazione **"Ha\_Presidente"** tra **Stato** e **Presidente**.
   * Molteplicità: Ogni Stato (repubblica) ha al massimo un Presidente.
3. **Stato-Monarca**:
   * Relazione **"Ha\_Monarca"** tra **Stato** e **Monarca**.
   * Molteplicità: Ogni Stato (monarchia) ha al massimo un Monarca.

-- Tabella Stato

CREATE TABLE Stato (

ID\_Stato INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(100) NOT NULL,

Continente VARCHAR(50) NOT NULL,

Numero\_Abitanti BIGINT NOT NULL,

Superficie FLOAT NOT NULL,

Densita FLOAT NOT NULL

);

-- Tabella Confinante

CREATE TABLE Confinante (

ID\_Stato1 INT NOT NULL,

ID\_Stato2 INT NOT NULL,

Lunghezza\_Confine FLOAT,

PRIMARY KEY (ID\_Stato1, ID\_Stato2),

CONSTRAINT FK\_Confinante\_Stato1 FOREIGN KEY (ID\_Stato1) REFERENCES Stato(ID\_Stato),

CONSTRAINT FK\_Confinante\_Stato2 FOREIGN KEY (ID\_Stato2) REFERENCES Stato(ID\_Stato)

);

-- Tabella Presidente

CREATE TABLE Presidente (

ID\_Presidente INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50) NOT NULL,

Cognome VARCHAR(50) NOT NULL,

Partito\_Politico VARCHAR(100),

Data\_Elezione DATE NOT NULL,

ID\_Stato INT NOT NULL,

CONSTRAINT FK\_Presidente\_Stato FOREIGN KEY (ID\_Stato) REFERENCES Stato(ID\_Stato)

);

-- Tabella Monarca

CREATE TABLE Monarca (

ID\_Monarca INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50) NOT NULL,

Cognome VARCHAR(50) NOT NULL,

Casato VARCHAR(100) NOT NULL,

ID\_Stato INT NOT NULL,

CONSTRAINT FK\_Monarca\_Stato FOREIGN KEY (ID\_Stato) REFERENCES Stato(ID\_Stato)

);

### ****Descrizione dello Schema Logico****

1. **Tabella Stato**:
   * Contiene le informazioni relative agli Stati: nome, continente, popolazione, superficie e densità.
2. **Tabella Confinante**:
   * Implementa la relazione **"Confina\_con"** tra Stati.
   * È una tabella autoreferenziale con due chiavi esterne (ID\_Stato1 e ID\_Stato2) che fanno riferimento alla tabella Stato.
3. **Tabella Presidente**:
   * Memorizza i dati dei presidenti degli Stati repubblicani.
   * Colonna ID\_Stato che stabilisce una relazione uno-a-uno con la tabella Stato.
4. **Tabella Monarca**:
   * Contiene i dettagli dei monarchi degli Stati monarchici.
   * Colonna ID\_Stato che stabilisce una relazione uno-a-uno con la tabella Stato.

Si vuole progettare una base di dati per gestire le informazioni su film e proiezioni: per le proiezioni è richiesto di memorizzare la città, la sala, la data, l’ora, il numero di spettatori per proiezione mentre per i film, che sono identificati da un codice, interessa il titolo, l’anno di produzione, il regista, gli attori che vi recitano e, per ogni attore, se è protagonista o non protagonista e la sua nazionalità. Fammi: 1. lo schema e-r e la sua immagine jpg 2. lo schema relazionale 3. lo schema logico cone le quesry sql per crearlo 4. una query di inserimento dati per ogni tabella 5. una pagina html e php con dei pulsanti: un pulsante per ogni tabella che visualizza i dati di ogni tabella separatamente

### ****1. Schema E-R (Entità-Relazione)****

### ****Entità e Attributi****

1. **Film**
   * Codice\_Film (Chiave primaria)
   * Titolo
   * Anno\_Produzione
   * Regista
2. **Attore**
   * Codice\_Attore (Chiave primaria)
   * Nome
   * Cognome
   * Nazionalità
   * Ruolo (Protagonista/Non Protagonista)
3. **Proiezione**
   * Codice\_Proiezione (Chiave primaria)
   * Città
   * Sala
   * Data
   * Ora
   * Numero\_Spettatori

### ****Relazioni****

1. **Film\_Attore**
   * Un film può avere molti attori e un attore può partecipare a molti film. (Relazione molti a molti)
   * Attributi: Ruolo (Protagonista o Non Protagonista)
2. **Film\_Proiezione**
   * Un film può avere molte proiezioni e ogni proiezione è associata a un solo film. (Relazione uno a molti)

Attore(Codice\_Attore PK, Nome, Cognome, Nazionalità)

Film(Codice\_Film PK, Titolo, Anno\_Produzione, Regista)

Proiezione(Codice\_Proiezione PK, Città, Sala, Data, Ora, Numero\_Spettatori)

Film\_Attore(Codice\_Film FK, Codice\_Attore FK, Ruolo)

Film\_Proiezione(Codice\_Film FK, Codice\_Proiezione FK)

-- Tabella Film

CREATE TABLE Film (

Codice\_Film INT PRIMARY KEY AUTO\_INCREMENT,

Titolo VARCHAR(100),

Anno\_Produzione INT,

Regista VARCHAR(100)

);

-- Tabella Attore

CREATE TABLE Attore (

Codice\_Attore INT PRIMARY KEY AUTO\_INCREMENT,

Nome VARCHAR(50),

Cognome VARCHAR(50),

Nazionalità VARCHAR(50)

);

-- Tabella Proiezione

CREATE TABLE Proiezione (

Codice\_Proiezione INT PRIMARY KEY AUTO\_INCREMENT,

Città VARCHAR(100),

Sala VARCHAR(50),

Data DATE,

Ora TIME,

Numero\_Spettatori INT

);

-- Tabella Film\_Attore (Relazione molti a molti tra Film e Attore)

CREATE TABLE Film\_Attore (

Codice\_Film INT,

Codice\_Attore INT,

Ruolo ENUM('Protagonista', 'Non Protagonista'),

PRIMARY KEY (Codice\_Film, Codice\_Attore),

FOREIGN KEY (Codice\_Film) REFERENCES Film(Codice\_Film),

FOREIGN KEY (Codice\_Attore) REFERENCES Attore(Codice\_Attore)

);

-- Tabella Film\_Proiezione (Relazione uno a molti tra Film e Proiezione)

CREATE TABLE Film\_Proiezione (

Codice\_Film INT,

Codice\_Proiezione INT,

PRIMARY KEY (Codice\_Film, Codice\_Proiezione),

FOREIGN KEY (Codice\_Film) REFERENCES Film(Codice\_Film),

FOREIGN KEY (Codice\_Proiezione) REFERENCES Proiezione(Codice\_Proiezione)

);

INSERT INTO Film (Titolo, Anno\_Produzione, Regista)

VALUES ('Il Signore degli Anelli', 2001, 'Peter Jackson');

INSERT INTO Attore (Nome, Cognome, Nazionalità)

VALUES ('Elijah', 'Wood', 'USA');

INSERT INTO Proiezione (Città, Sala, Data, Ora, Numero\_Spettatori)

VALUES ('Roma', 'Sala 1', '2025-02-15', '20:00:00', 150);

INSERT INTO Film\_Attore (Codice\_Film, Codice\_Attore, Ruolo)

VALUES (1, 1, 'Protagonista');

INSERT INTO Film\_Proiezione (Codice\_Film, Codice\_Proiezione)

VALUES (1, 1);

La riga di PHP:

$result = $conn->query("SELECT \* FROM Film");

è una parte importante del codice che si occupa di **eseguire una query SQL** sul database per ottenere i dati dalla tabella **Film**. Vediamo nel dettaglio il significato di ogni parte di questa riga:

### 1. $conn

* $conn è una variabile che rappresenta l'oggetto di connessione al database.
* Questo oggetto è stato creato precedentemente con la riga di codice:

php

Copia codice

$conn = new mysqli('localhost', 'root', '', 'film\_db');

* In questa riga, $conn è un'istanza della classe mysqli che permette di interagire con il database MySQL. Con questa connessione, puoi eseguire query, recuperare dati, e fare altre operazioni sul database.

### 2. ->

* L'operatore -> in PHP è usato per accedere a un metodo o una proprietà di un oggetto. In questo caso, accediamo al metodo query() dell'oggetto $conn.
* In PHP, gli oggetti sono istanze di una classe, e l'operatore -> viene usato per chiamare metodi e accedere alle proprietà di un oggetto.

### 3. query()

* Il metodo query() è un metodo della classe mysqli che viene usato per eseguire una query SQL sul database.
* Questo metodo prende come argomento una stringa contenente una query SQL.
* La query SQL può essere qualsiasi comando supportato da MySQL, come SELECT, INSERT, UPDATE, DELETE, ecc.

In questo caso, stiamo eseguendo una query SELECT che serve a recuperare dei dati.

### 4. "SELECT \* FROM Film"

* La stringa "SELECT \* FROM Film" è la query SQL che verrà eseguita sul database.
* **SELECT** è una dichiarazione SQL usata per selezionare i dati da una o più tabelle.
* **\*** è un operatore che indica "tutti i campi". Quindi, con SELECT \*, stiamo dicendo al database di restituire **tutti i campi** (colonne) dalla tabella **Film**.
* **FROM Film** indica da quale tabella vogliamo selezionare i dati. In questo caso, stiamo selezionando dalla tabella **Film**.

Quindi, la query SQL completa significa: "Seleziona tutte le colonne da tutti i record della tabella *Film*."

### 5. $result

* Il risultato della query viene memorizzato nella variabile $result.
* Il valore che $result contiene dipende dal tipo di query:
  + Se la query è un SELECT, $result sarà un oggetto di tipo mysqli\_result, che contiene tutte le righe e le colonne restituite dalla query.
  + Se la query non restituisce risultati (ad esempio, una INSERT o DELETE), $result sarà un valore booleano TRUE o FALSE.
* Nel caso di un SELECT, $result sarà un oggetto che può essere utilizzato per ottenere i dati riga per riga, come nell'esempio seguente:

php

Copia codice

while ($row = $result->fetch\_assoc()) {

echo $row['Titolo'];

}

### ****Riepilogo****:

Quindi, la riga di codice:

php

$result = $conn->query("SELECT \* FROM Film");

fa quanto segue:

1. Esegue una query SQL per selezionare tutti i record (righe) e tutte le colonne dalla tabella Film.
2. Memorizza il risultato della query nell'oggetto $result.
3. Puoi usare $result per ottenere i dati selezionati dalla tabella, come ad esempio iterare sulle righe usando fetch\_assoc() per recuperare i valori delle colonne.

 **$result = $conn->query("SELECT \* FROM Film");**

* Questa riga esegue una query SQL sulla tabella Film del database e memorizza il risultato nella variabile $result.
* La query seleziona tutte le righe della tabella Film (indicato con SELECT \*).

 **while ($row = $result->fetch\_assoc())**

* Questo è un ciclo while che itera su ogni riga restituita dalla query.
* $result->fetch\_assoc() è una funzione che recupera una riga di risultato dalla query sotto forma di array associativo, dove ogni chiave dell'array corrisponde al nome della colonna della tabella e il valore associato è il dato contenuto in quella colonna per quella riga.
* La variabile $row memorizza ogni singola riga (array associativo) durante ogni iterazione del ciclo. Il ciclo continua finché ci sono altre righe da elaborare.

 **Dentro il ciclo while**:

* echo "<tr><td>" . $row['Codice\_Film'] . "</td><td>" . $row['Titolo'] . "</td><td>" . $row['Anno\_Produzione'] . "</td><td>" . $row['Regista'] . "</td></tr>";
* Qui, per ogni riga $row, viene creato un nuovo elemento di tabella (<tr>) con i valori delle colonne Codice\_Film, Titolo, Anno\_Produzione, e Regista che vengono recuperati dall'array associativo $row.
* Ogni valore di colonna viene inserito in una cella di tabella (<td>) corrispondente.

 **Fine del ciclo while**:

* Il ciclo continua finché fetch\_assoc() non restituisce false, il che significa che non ci sono più righe da elaborare.