

*Ministero dell'istruzione e del merito***A038 - ESAME DI STATO CONCLUSIVO DEL SECONDO CICLO DI ISTRUZIONE****Indirizzo ITIA - INFORMATICA E TELECOMUNICAZIONI ARTICOLAZIONE "INFORMATICA"****(Testo valevole anche per gli indirizzi quadriennali IT32 e ITIT)****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;


Ministero dell'istruzione e del merito
A038 - ESAME DI STATO CONCLUSIVO DEL SECONDO CICLO DI ISTRUZIONE

Indirizzo ITIA - INFORMATICA E TELECOMUNICAZIONI ARTICOLAZIONE "INFORMATICA"
(Testo valevole anche per gli indirizzi quadriennali IT32 e ITIT)

Disciplina: INFORMATICA

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

Durata massima della prova: 6 ore.

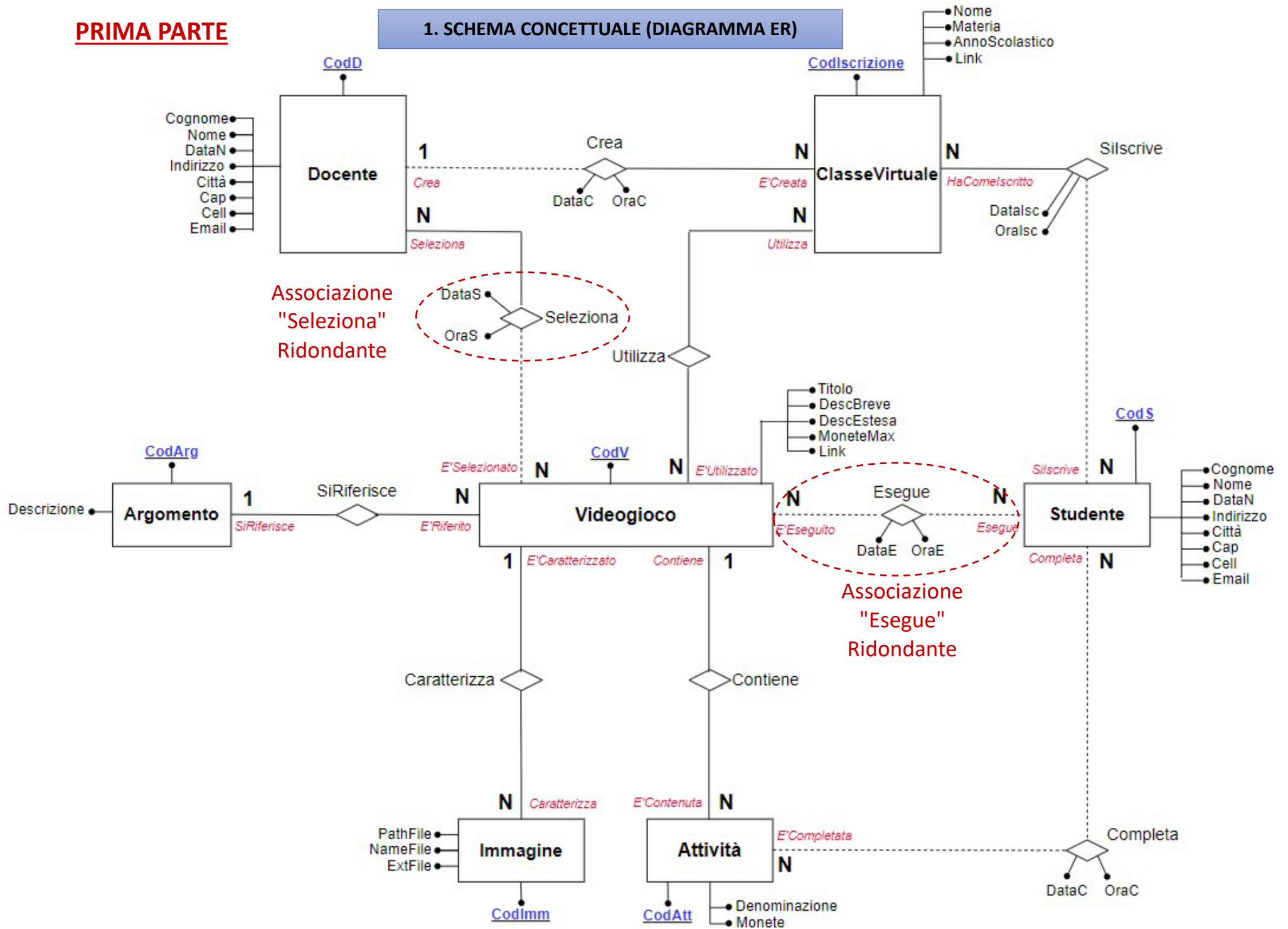
È consentito soltanto l'uso dei manuali di riferimento dei linguaggi di programmazione (language reference) e di calcolatrici scientifiche e/o grafiche purché non siano dotate di capacità di calcolo simbolico.

È consentito l'uso del dizionario bilingue (italiano-lingua del paese di provenienza) per i candidati di madrelingua non italiana.

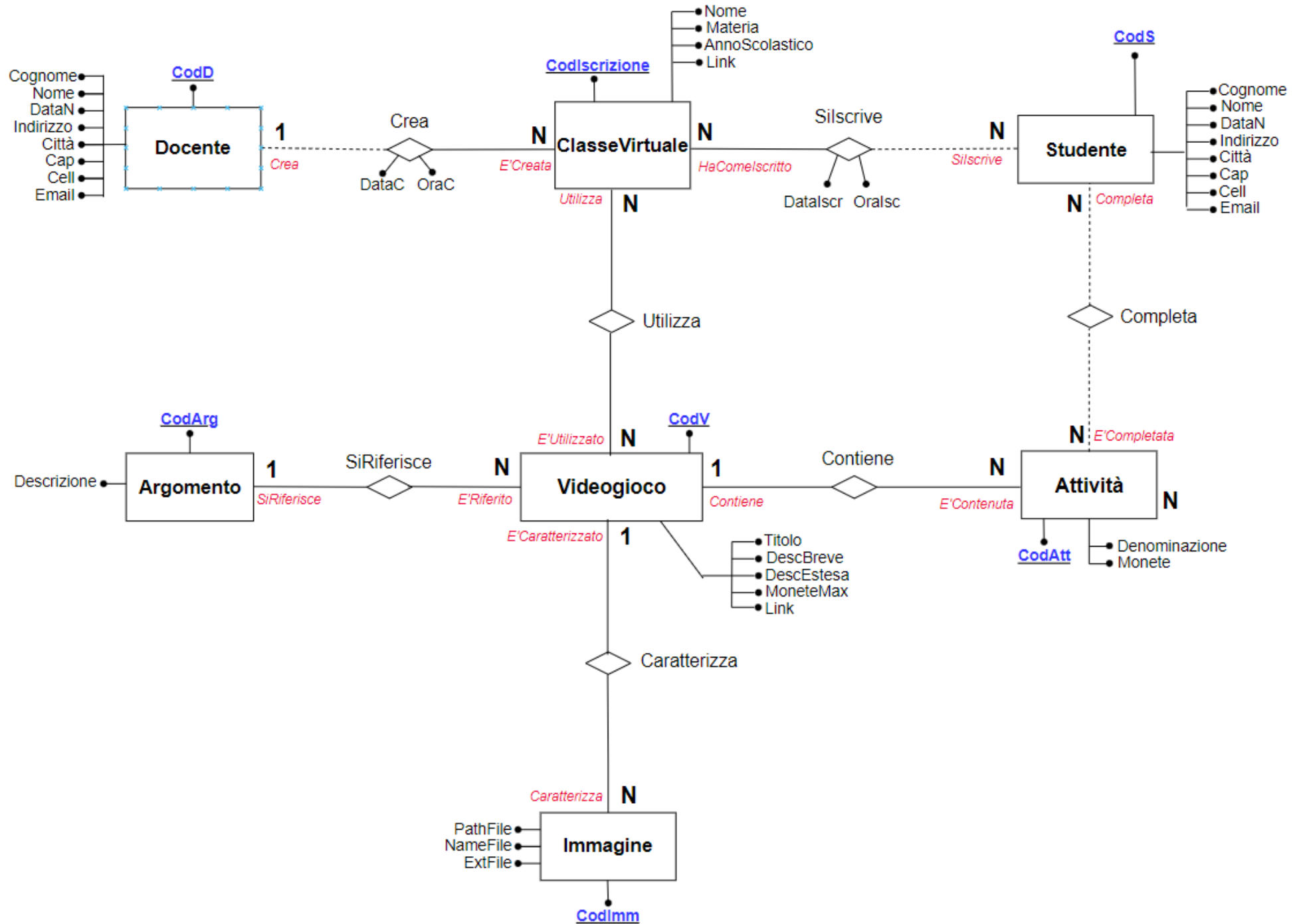
Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla consegna della traccia.

PRIMA PARTE

1. SCHEMA CONCETTUALE (DIAGRAMMA ER)



1. SCHEMA CONCETTUALE (DIAGRAMMA ER)



Ipotesi aggiuntive

- 1) Si è preferito, per ragioni di efficienza, stabilire come chiave primaria (PK) dell'entità "ClasseVirtuale", l'**unico** attributo "CodIscrizione" piuttosto che la coppia di attributi "Nome" e "Materia";
- 2) L'attributo "MoneteMax" dell'entità "Videogioco" dovrà essere valorizzato, per ragioni di consistenza, aggiungendo di volta in volta il valore contenuto nell'attributo "Monete" di ciascuna "Attività" che esso conterrà.

Regole di lettura

- a) Un **argomento** deve riferirsi ad uno o più **videogiochi** e viceversa un **videogioco** deve essere riferito ad un ed un solo **argomento**
- b) Un **docente** può creare nessuna o più **classi virtuali** e viceversa una **classe virtuale** deve essere creata da uno ed un solo **docente**
- c) Una **classe virtuale** deve utilizzare uno o più **videogiochi** e viceversa un **videogioco** deve essere utilizzato in una o più **classi virtuali**
- d) Uno **studente** può iscriversi a nessuna o più **classi virtuali** e viceversa una **classe virtuale** deve avere come iscritti uno o più **studenti**
- e) Uno **studente** può completare nessuna o più **attività** e viceversa una **attività** può essere completata da nessuno o più **studenti**
- f) Un **videogioco** deve contenere una o più **attività** e viceversa una **attività** deve essere contenuta da uno ed un solo **videogioco**
- g) Una **immagine** deve caratterizzare uno ed un solo **videogioco** e viceversa un **videogioco** deve essere caratterizzato da una o più **immagini** (N.B. max 3)

Vincoli di Integrità

I vincoli di integrità (V.d.I) si dividono in:

- Vincoli di integrità **IMPLICITI**: a loro volta si dividono in:

- V.d.I. **IMPLICITI** dovuti alla **chiave primaria o primary key (PK)**:

- l'attributo "CodArg" è la primary key (PK) dell'entità "Argomento"
- l'attributo "CodD" è la primary key (PK) dell'entità "Docente"
- l'attributo "CodIscrizione" è la primary key (PK) dell'entità "ClasseVirtuale"
- l'attributo "CodS" è la primary key (PK) dell'entità "Studente"
- l'attributo "CodAtt" è la primary key (PK) dell'entità "Attività"
- l'attributo "CodImm" è la primary key (PK) dell'entità "Immagine"
- l'attributo "CodV" è la primary key (PK) dell'entità "Videogioco"

- V.d.I. **IMPLICITI** dovuti alla **TOTALITA'** delle associazioni DIRETTE e/o INVERSE;

- TOTALITA' dell'ASS. DIRETTA "SiRiferisce" tra le entità "Argomento" e "Videogioco"
- TOTALITA' dell'ASS. INVERSA "E'Riferito" tra le entità "Videogioco" e "Argomento"
- TOTALITA' dell'ASS. INVERSA "E'Creata" tra le entità "ClasseVirtuale" e "Docente"
- TOTALITA' dell'ASS. DIRETTA "Include" tra le entità "ClasseVirtuale" e "Videogioco"
- TOTALITA' dell'ASS. INVERSA "E'Incluso" tra le entità "Videogioco" e "ClasseVirtuale"
- TOTALITA' dell'ASS. INVERSA "HaComelscritto" tra le entità "ClasseVirtuale" e "Studente"
- TOTALITA' dell'ASS. DIRETTA "Contiene" tra le entità "Videogioco" e "Attività"
- TOTALITA' dell'ASS. INVERSA "E'Contenuta" tra le entità "Attività" e "Videogioco"
- TOTALITA' dell'ASS. DIRETTA "Caratterizza" tra le entità "Immagine" e "Videogioco"
- TOTALITA' dell'ASS. INVERSA "E'Caratterizzato" tra le entità "Videogioco" e "Immagine"

- V.d.I. **ESPLICITI**

- **V1** : (Attività.Monete BETWEEN 1 AND 10)
- **V2** : (Docente.DataN < Studente.DataN)
- **V3** : (Silscrive.DataNsc > Studente.DataN)

2. SCHEMA LOGICO-RELAZIONALE

a) Mapping logico relazionale dell'associazione binaria "SiRiferisce" tra le entità "Argomento" e "Videogioco" di molteplicità 1:N

Argomento (CodArg, Descrizione)

Videogioco (CodV, Titolo, DescBreve, DescEstesa, MoneteMax, Link, CodArg1)

con l'attributo "CodArg1" della relazione "Videogioco" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodArg" della relazione "Argomento"

$VR_{CodArg}(\text{Argomento}) \subseteq VR_{CodArg1}(\text{Videogioco})$ Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "DIRETTA "SiRiferisce"

$VR_{CodArg1}(\text{Videogioco}) \subseteq VR_{CodArg}(\text{Argomento})$ Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "INVERSA "E'Riferito"

b) Mapping logico relazionale dell'associazione binaria "Crea" tra le entità "Docente" e "ClasseVirtuale" di molteplicità 1:N

Docente : relazione già mappata in precedenza

ClasseVirtuale (CodIscrizione, Nome, Materia, AnnoScolastico, Link, DataC, OraC, CodD2)

con l'attributo "CodD2" della relazione "ClasseVirtuale" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodD" della relazione "Docente"

$VR_{CodD2}(\text{ClasseVirtuale}) \subseteq VR_{CodD}(\text{Docente})$ Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "INVERSA "E'Creata"

c) Mapping logico relazionale dell'associazione binaria "Utilizza" tra le entità "ClasseVirtuale" e "Videogioco" di molteplicità N:N

ClasseVirtuale : relazione già mappata in precedenza

Videogioco : relazione già mappata in precedenza

Utilizza (CodIscrizione2, CodV2)

N.B. SENZA gestione dello storico

con l'attributo "CodIscrizione2" della relazione "Utilizza" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodIscrizione" della relazione "ClasseVirtuale"

con l'attributo "CodV2" della relazione "Utilizza" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodV" della relazione "Videogioco"

$VR_{CodIscrizione2}(\text{Include}) \subseteq VR_{CodIscrizione}(\text{ClasseVirtuale})$ Vincoli Referenziali (VR) causati dal mapping relazionale di una generica associazione di molteplicità N:N

$VR_{CodV2}(\text{Include}) \subseteq VR_{CodV}(\text{Videogioco})$

$VR_{CodIscrizione}(\text{ClasseVirtuale}) \subseteq VR_{CodIscrizione2}(\text{Include})$ Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "DIRETTA "Utilizza"

$VR_{CodV}(\text{Videogioco}) \subseteq VR_{CodV2}(\text{Include})$ Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "INVERSA "E'Utilizzato"

d) Mapping logico relazionale dell'associazione binaria "SiIscrive" tra le entità "Studente" e "ClasseVirtuale" di molteplicità N:N

Studente (CodS, Cognome, Nome, DataN, Indirizzo, Città, Cap, Cell, Email)

ClasseVirtuale : relazione già mappata in precedenza

SiIscrive (IdS1, CodS1, CodIscrizione1, DataIsc, OraIsc)

N.B. CON gestione dello storico

con l'attributo "CodS1" della relazione "Silscrive" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodS" della relazione "Studente"

con l'attributo "CodIscrizione1" della relazione "Silscrive" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodIscrizione" della relazione "ClasseVirtuale"

$$\left\{ \begin{array}{l} \text{VR}_{\text{CodS1}} (\text{Silscrive}) \subseteq \text{VR}_{\text{CodS}} (\text{Studente}) \\ \text{VR}_{\text{CodIscrizione1}} (\text{Silscrive}) \subseteq \text{VR}_{\text{CodIscrizione}} (\text{ClasseVirtuale}) \end{array} \right. \quad \text{Vincoli Referenziali (VR) causati dal mapping relazionale di una generica associazione di molteplicità N:N}$$

$\text{VR}_{\text{CodIscrizione}} (\text{ClasseVirtuale}) \subseteq \text{VR}_{\text{CodIscrizione1}} (\text{Silscrive})$ Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "INVERSA "HaComelscritto"

e) Mapping logico relazionale dell'associazione binaria "**Completa**" tra le entità "**Studente**" e "**Attività**" di molteplicità **N:N**

Studente : relazione già mappata in precedenza

Attività (CodAtt, Denominazione, Monete, CodV3)

con l'attributo "CodV3" della relazione "Attività" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodV" della relazione "Videogioco"

Completa (IdC, CodS4, CodAtt4, DataC, OraC)

N.B. CON gestione dello storico

con l'attributo "CodS4" della relazione "Completa" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodS" della relazione "Studente"

con l'attributo "CodAtt4" della relazione "Completa" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodAtt" della relazione "Attività"

$$\left\{ \begin{array}{l} \text{VR}_{\text{CodS4}} (\text{Completa}) \subseteq \text{VR}_{\text{CodS}} (\text{Studente}) \\ \text{VR}_{\text{CodAtt4}} (\text{Completa}) \subseteq \text{VR}_{\text{CodAtt}} (\text{Attività}) \end{array} \right. \quad \text{Vincoli Referenziali (VR) causati dal mapping relazionale di una generica associazione di molteplicità N:N}$$

f) Mapping logico relazionale dell'associazione binaria "**Contiene**" tra le entità "**Videogioco**" e "**Attività**" di molteplicità **1:N**

Videogioco : relazione già mappata in precedenza

Attività : relazione già mappata in precedenza

$\text{VR}_{\text{CodV}} (\text{Videogioco}) \subseteq \text{VR}_{\text{CodV3}} (\text{Attività})$

Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "DIRETTA "Contiene"

$\text{VR}_{\text{CodV3}} (\text{Attività}) \subseteq \text{VR}_{\text{CodV}} (\text{Videogioco})$

Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "INVERSA "E'Contenuta"

g) Mapping logico relazionale dell'associazione binaria "**Caratterizza**" tra le entità "**Immagine**" e "**Videogioco**" di molteplicità **N:1**

Immagine (CodImm, PathFile, NameFile, ExtFile, CodV4)

con l'attributo "CodV4" della relazione "Immagine" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodV" della relazione "Videogioco"

Videogioco : relazione già mappata in precedenza

$\text{VR}_{\text{CodV4}} (\text{Immagine}) \subseteq \text{VR}_{\text{CodV}} (\text{Videogioco})$

Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "DIRETTA "Caratterizza"

$\text{VR}_{\text{CodV}} (\text{Videogioco}) \subseteq \text{VR}_{\text{CodV4}} (\text{Immagine})$

Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione "INVERSA "E'Caratterizzato"

h) Mapping logico relazionale dei vincoli di integrità (V.d.I.)

<u>Diagramma ER</u>		<u>Schema logico relazionale</u>
V. d. I. impliciti di chiave primaria	⇒	Vincoli intarrelazionali o interni su più n-ple
V. d. I. impliciti dovuti alla totalità delle associazioni dirette e/o inverse	⇒	Vincoli interrelazionali o esterni referenziali
V1 : (.....)	⇒	V1 (Attività) : (Monete BETWEEN 1 AND 10) Vincolo intrarelazionale o interno su singola n-ple sul dominio di un attributo
V2 : (.....)	⇒	V2 (Docente, Studente) : (Docente.DataN < Studente.DataN) Vincolo interrelazionale o esterno NON REFERENZIALE
V3 : (.....)	⇒	V3 (Silscrive, Studente) : (Silscrive.DataIsc > Studente.DataN) Vincolo interrelazionale o esterno NON REFERENZIALE

A tutti questi vincoli di integrità vanno aggiunti tutti i vincoli di integrità referenziali dovuti al mapping relazionale di una generica associazione di molteplicità N:N che sono mappati come **vincoli interrelazionali o esterni REFERENZIALI**.

Il nostro modello logico relazionale risulta, per le scelte implementative fatte, in terza forma normale o 3FN

3. SCHEMA FISICO (RELAZIONI IN LINGUAGGIO SQL)

Premessa

Supponiamo che il DBA (Database Administrator) o altro utente con medesimi privilegi abbia creato un utente dotato di password che possenga tutti i principali permessi (CREATE, DROP, ALTER, SELECT, INSERT, UPDATE, DELETE, etc.) su tutte le tabelle del database che andremo a creare.

Supponiamo quindi che tale utente abbia eseguito il login ed effettuato le seguenti query in modo interattivo o utilizzando la modalità embedded.

```
CREATE DATABASE EduVideogame;
```

```
USE EduVideogame;
```

```
CREATE TABLE Argomento
```

```
(  
  CodArg    VARCHAR (10) NOT NULL,  
  Descrizione VARCHAR (150) NOT NULL,  
  PRIMARY KEY (CodArg)  
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;
```

```
CREATE TABLE Videogioco
```

```
(  
  CodV      VARCHAR(10) NOT NULL,  
  Titolo    VARCHAR (30) NOT NULL,  
  DescBreve VARCHAR (160) NOT NULL,  
  DescEstesa VARCHAR (2000) NOT NULL,      # VARCHAR fino a 65535 caratteri altrimenti BLOB o TEXT  
  MoneteMax INT(3) NOT NULL DEFAULT 0,    # max 99 monete virtuali raggiungibili  
  Link      VARCHAR (255) NOT NULL,  
  CodArg1   VARCHAR (10) NOT NULL,  
  UNIQUE (Link),  
  PRIMARY KEY (CodV),  
  FOREIGN KEY (CodArg1) REFERENCES Argomento (CodArg)  
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;
```

```
CREATE TABLE Docente
```

```
(  
  CodD      VARCHAR(10) NOT NULL,  
  Cognome   VARCHAR (50) NOT NULL,  
  Nome      VARCHAR(50) NOT NULL,  
  DataN     DATE NOT NULL,  
  Citta     VARCHAR (30) NOT NULL,        # N.B. Tolto il carattere accentato per non avere problemi  
  Cap      CHAR(5) NOT NULL,  
  Cell      VARCHAR(20) NOT NULL,  
  Email     VARCHAR(50) NOT NULL,  
  UNIQUE (Cell),  
  UNIQUE (Email),  
  PRIMARY KEY (CodD)  
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;
```

```
CREATE TABLE ClasseVirtuale
```

```
(  
  CodIscrizione VARCHAR(10) NOT NULL,  
  Nome           CHAR (2) NOT NULL,  
  Materia        VARCHAR (50) NOT NULL,  
  AnnoScolastico CHAR (9) NOT NULL,      #tipo 2022/2023  
  Link           VARCHAR (255) NOT NULL,  
  DataC          DATE NOT NULL,
```

```

OraC          TIME NOT NULL,
CodD2         VARCHAR (10) NOT NULL,
UNIQUE (Link),
UNIQUE (Nome, Materia),
PRIMARY KEY (CodIscrizione),
FOREIGN KEY (CodD2) REFERENCES Docente (CodD)
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;

```

CREATE TABLE Utilizza # N.B. SENZA gestione dello storico

```

(
CodIscrizione2  VARCHAR (10) NOT NULL,
CodV2          VARCHAR (10) NOT NULL,
PRIMARY KEY (CodIscrizione2, CodV2),
FOREIGN KEY (CodIscrizione2) REFERENCES ClasseVirtuale (CodIscrizione)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (CodV2) REFERENCES VideoGioco (CodV)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;

```

CREATE TABLE Studente

```

(
CodS          VARCHAR(10) NOT NULL,
Cognome       VARCHAR (50) NOT NULL,
Nome          VARCHAR(50) NOT NULL,
DataN         DATE NOT NULL,
Citta         VARCHAR (30) NOT NULL,
Cap           CHAR(5) NOT NULL,
Cell          VARCHAR(20) NOT NULL,
Email         VARCHAR(50) NOT NULL,
UNIQUE (Cell),
UNIQUE (Email),
PRIMARY KEY (CodS)
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;

```

N.B. Tolto il carattere accentato per non avere problemi

CREATE TABLE Silscrive # N.B. CON gestione dello storico

```

(
IdS1          INT(5) NOT NULL AUTO_INCREMENT,
CodS1         VARCHAR (10) NOT NULL,
CodIscrizione1  VARCHAR (10) NOT NULL,
DataIsc       DATE NOT NULL,
Oralsc        TIME NOT NULL,
PRIMARY KEY (IdS1),
FOREIGN KEY (CodS1) REFERENCES Studente (CodS)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (CodIscrizione1) REFERENCES ClasseVirtuale (CodIscrizione)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;

```

CREATE TABLE Attivita # N.B. Tolto il carattere accentato per non avere problemi

```
(
CodAtt          VARCHAR (10) NOT NULL,
Denominazione  VARCHAR (50) NOT NULL,
Monete         INT (2)  NOT NULL,
CodV3          VARCHAR (10) NOT NULL,
PRIMARY KEY (CodAtt),
FOREIGN KEY (CodV3) REFERENCES Videogioco (CodV),
CHECK (Monete BETWEEN 1 AND 10) # Vincolo V1
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;
```

CREATE TABLE Completa # N.B. CON gestione dello storico

```
(
IdC             INT(5) NOT NULL AUTO_INCREMENT,
CodS4          VARCHAR (10) NOT NULL,
CodAtt4       VARCHAR (10) NOT NULL,
DataC         DATE NOT NULL,
OraC          TIME NOT NULL,
PRIMARY KEY (IdC),
FOREIGN KEY (CodS4) REFERENCES Studente (CodS)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
FOREIGN KEY (CodAtt4) REFERENCES Attivita (CodAtt)
  ON DELETE CASCADE
  ON UPDATE CASCADE
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;
```

CREATE TABLE Immagine

```
(
CodImm        VARCHAR (10) NOT NULL,
PathFile     VARCHAR (200) NOT NULL,
NameFile     VARCHAR (50) NOT NULL,
CodV4        VARCHAR (10) NOT NULL,
PRIMARY KEY (CodImm),
FOREIGN KEY (CodV4) REFERENCES Videogioco (CodV)
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;
```

CREATE ASSERTION V2 CHECK (Docente.DataN < Studente.DataN); # Vincolo V2

CREATE ASSERTION V3 CHECK (Silscribe.DataIsc > Studente.DataN); # Vincolo V3

```
#TRIGGER MoneteINMoneteMax1: per controllare che il valore dell'attributo "MoneteMax" della tabella
# "Videogioco" non superi il valore massimo previsto (999) per ogni inserzione di una attività che si
# riferisce a lui
```

```
DELIMITER //
CREATE TRIGGER MoneteINMoneteMax1
BEFORE INSERT ON EduVideogame.Attivita
FOR EACH ROW
BEGIN
DECLARE msg VARCHAR(255);
DECLARE x INT;
SET x = (SELECT MoneteMax FROM EduVideogame.Videogioco WHERE CodV = NEW.CodV3);
IF x + NEW.Monete > 999
THEN
SET msg = concat("L'attributo 'MoneteMax' non puo' superare il valore 999");
SIGNAL sqlstate '45000' SET message_text = msg;
END IF;
END //
DELIMITER ;
```

```
#TRIGGER MoneteINMoneteMax2: per aumentare il valore dell'attributo "MoneteMax" della tabella
# "Videogioco" del valore dello attributo "Monete" per ogni inserzione di una attività che
# si riferisce a lui
```

```
DELIMITER //
CREATE TRIGGER MoneteINMoneteMax2
AFTER INSERT ON EduVideogame.Attivita
FOR EACH ROW
BEGIN
DECLARE msg VARCHAR(255);
IF NEW.Monete >= 1 AND NEW.Monete <= 10
THEN
UPDATE EduVideogame.Videogioco
SET MoneteMax = MoneteMax + NEW.Monete
WHERE CodV = NEW.CodV3;
ELSE
SET msg = concat("L'attributo 'Monete' deve assumere un valore tra 1 e 10!");
SIGNAL sqlstate '45000' SET message_text = msg;
END IF;
END //
DELIMITER ;
```

4. QUERY (IN LINGUAGGIO SQL)

a) l'elenco in ordine alfabetico dei giochi classificati per uno specifico argomento;

```
SELECT Videogioco.Titolo, Videogioco.DescBreve
FROM Argomento, Videogioco
WHERE (Argomento.CodArg = Videogioco.CodArg1)
      AND (Argomento.Descrizione = [DescrizioneX])
ORDER BY Titolo ASC;
```

esempio con dati

```
SELECT Videogioco.Titolo, Videogioco.DescBreve
FROM Argomento, Videogioco
WHERE (Argomento.CodArg = Videogioco.CodArg1)
      AND (Argomento.Descrizione = "GLI ARRAY MONODIMENSIONALI")
ORDER BY Titolo ASC;
```

b) la classifica degli studenti di una certa classe virtuale, in base alle monete raccolte per un certo gioco;

n.b. SCELTA IMPLEMENTATIVA PER ESEGUIRE MENO JOIN:

la classe virtuale è identificata dal codice presente sulla tabella **Silscrive**

il videogioco è identificato dal codice presente sulla tabella **Attivita**

```
SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
FROM Silscrive, Studente, Completa, Attivita
WHERE (Silscrive.CodS1 = Studente.CodS) AND (Studente.CodS = Completa.CodS4)
      AND (Completa.CodAtt4 = Attivita.CodAtt)
      AND (Silscrive.CodIscrizione1 = [Cod_ClasseX])
      AND (Attivita.CodV3 = [Cod_VideogiocoX])
GROUP BY Studente.Cognome, Studente.Nome
ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
```

esempio con dati

```
SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
FROM Silscrive, Studente, Completa, Attivita
WHERE (Silscrive.CodS1 = Studente.CodS) AND (Studente.CodS = Completa.CodS4)
      AND (Completa.CodAtt4 = Attivita.CodAtt)
      AND (Silscrive.CodIscrizione1 = "AXmm04")
      AND (Attivita.CodV3 = "V-001")
GROUP BY Studente.Cognome, Studente.Nome
ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
```

SE SI DESIDERASSE utilizzate l'attributo **Titolo** per il **videogioco** e

gli attributi **Nome** e **Materia** per la **classe virtuale** occorrerebbe aggiungere le relative tabelle

aggiungendo ulteriori condizioni di join

```
SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
FROM Studente, Silscrive, ClasseVirtuale, Completa, Attivita, Videogioco
WHERE (Studente.CodS = Silscrive.CodS1) AND (Silscrive.CodIscrizione1 = ClasseVirtuale.CodIscrizione)
      AND (Studente.CodS = Completa.CodS4) AND (Completa.CodAtt4 = Attivita.CodAtt)
      AND (Attivita.CodV3 = Videogioco.CodV)
      AND (ClasseVirtuale.Nome = [NomeX]) AND (ClasseVirtuale.Materia = [MateriaX])
      AND (Videogioco.Titolo = [TitoloX])
GROUP BY Studente.Cognome, Studente.Nome
ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
```


esempio con dati

```
SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
FROM Studente, Silscrive, ClasseVirtuale, Completa, Attivita, Videogioco
WHERE (Studente.CodS = Silscrive.CodS1) AND (Silscrive.CodIscrizione1 = ClasseVirtuale.CodIscrizione)
AND (Studente.CodS = Completa.CodS4) AND (Completa.CodAtt4 = Attivita.CodAtt)
AND (Attivita.CodV3 = VideoGioco.CodV)
AND (ClasseVirtuale.Nome = "3H") AND (ClasseVirtuale.Materia = "INFORMATICA")
AND (Videogioco.Titolo = "Ordinamento")
GROUP BY Studente.Cognome, Studente.Nome
ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
```

c) il numero di classi in cui è utilizzato ciascun videogioco del catalogo;

```
SELECT Utilizza.CodV2, COUNT(*) AS TotaleClassi # n.b. videogioco identificato dal suo Codice
FROM Utilizza, Videogioco
WHERE (Utilizza.CodV2 = Videogioco.CodV)
GROUP BY Utilizza.CodV2;
```

oppure

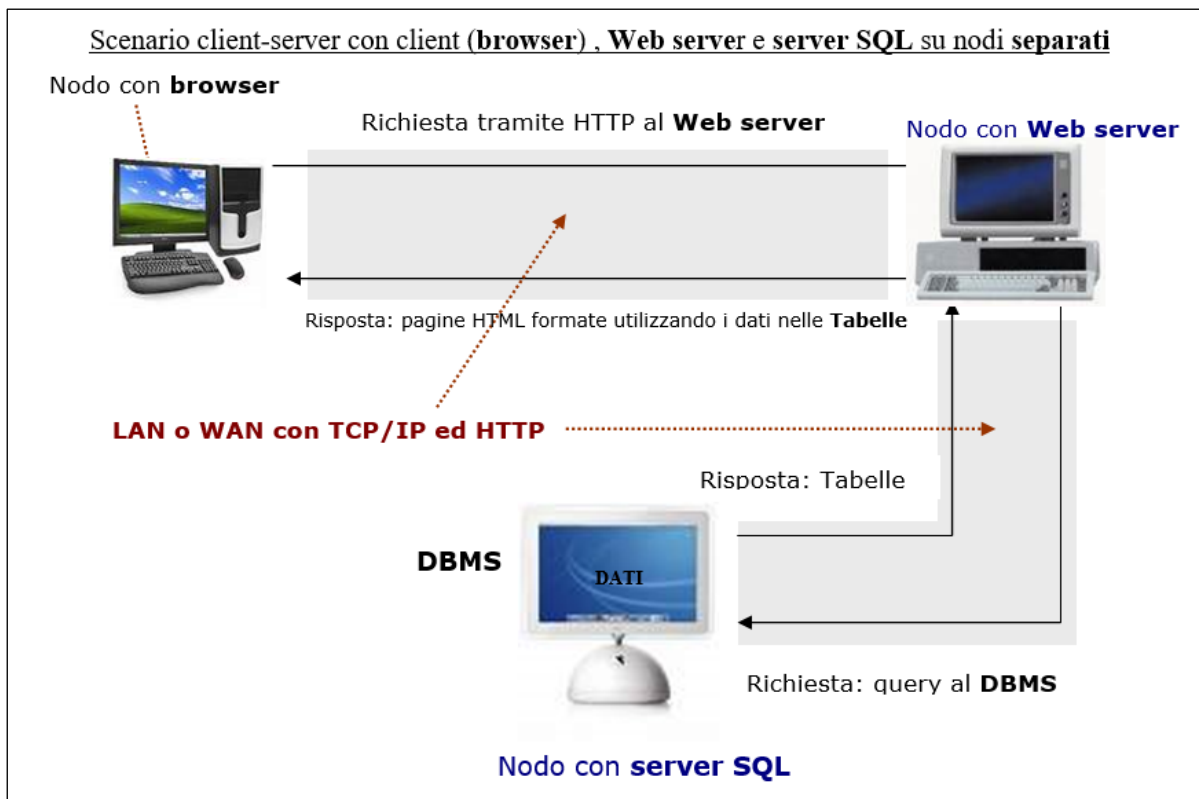
```
SELECT Videogioco.Titolo, COUNT(*) AS TotaleClassi # n.b. videogioco identificato dal suo Titolo
FROM ClasseVirtuale, Utilizza, Videogioco
WHERE (ClasseVirtuale.CodIscrizione = Utilizza.CodIscrizione2) AND
(Utilizza.CodV2 = Videogioco.CodV)
GROUP BY Videogioco.Titolo;
```

5. PROGETTO (di massima) STRUTTURA APPLICAZIONE WEB

Il nostro obiettivo è utilizzare l'applicazione web richiesta in un **ambiente di rete** ossia quello di far interagire un **database** situato fisicamente su un **nodo server** con i **browser** situati su qualsiasi altro **nodo client** della **rete** (Internet o Intranet)

Quindi siamo nell'ambito della **programmazione orientata al Web o Web-oriented** termine con il quale si intende l'insieme di tecniche e metodologie che si possono usare in un *ambiente client-server* con un'architettura di *protocolli TCP/IP e HTTP* per far interagire tra loro *programmi lato server* e *programmi lato client* con l'obiettivo di realizzare sistemi che possono essere eseguiti in una *Intranet* oppure in *Internet*.

Sono possibili diversi scenari: quello che viene proposto qui è di carattere generale e prevede il seguente schema esemplificativo



dove:

- con **Web server** si intende un software (e, per estensione, il computer) che si occupa di fornire, su richiesta del **browser** una pagina web (spesso scritta in HTML) della nostra applicazione. Le informazioni inviate dal Web server viaggiano in rete trasportate dal protocollo HTTP. L'insieme dei Web server presenti su Internet forma il WWW ossia il World Wide Web, uno dei servizi più sfruttati della Grande Rete (nel nostro caso **APACHE**);
- con **Database server** si intende un software (e, per estensione, il computer) che permette l'accesso ad altri programmi della Rete, ad uno o più Sistemi di DataBase ed è un supporto essenziale, per i Web Server, nel gestire la Consegna e la Memorizzazione di Dati (nel nostro caso **MySQL**).

Per realizzare gli script della nostra applicazione utilizzeremo la **programmazione lato server** (utilizzando il **linguaggio PHP**) ossia svilupperemo le pagine dell'applicazione richiesta che andranno in esecuzione prevalentemente sul *server*, accettando le richieste dal client (browser) e fornendo a quest'ultimo i risultati dell'elaborazione sottoforma di pagine HTML.

6. INTERAZIONE HTML-PHP-MYSQL

Layout FORM HTML: **query_a.htm**

Query a

Descrizione argomento:

Codice HTML

```
<HTML>
  <HEAD>
    <TITLE>ESAME DI STATO 2022/2023 - Query A</TITLE>
  </HEAD>
  <BODY>
    <FORM name="form" action="query_a.php" method="POST" target="_SELF">
      <FIELDSET>
        <LEGEND>Query a</LEGEND>
        <LABEL>Descrizione argomento:</LABEL>
        <INPUT TYPE="text" name="Descrizione" size="50" maxlength="150"
          value="GLI ARRAY MONODIMENSIONALI" required ><BR><BR>
        <INPUT TYPE="submit" value="Esegui">
        <INPUT TYPE="reset" value="Reset">
      </FIELDSET>
    </FORM>
  </BODY>
</HTML>
```

SCRIPT PHP-MySQL: **query_a.php**

```
<HTML>
  <HEAD>
    <TITLE>ESAME DI STATO 2022/2023 - Query A</TITLE>
  </HEAD>
  <BODY>
    <?php
    // Acquisizione parametri per connessione
    include ("include\parameter.php");
    //Apertura connessione al database
    $conn = mysqli_connect($host, $utente, $pswd, $db) OR
      die("Connessione NOT OK! ".mysqli_connect_error()." ".mysqli_connect_errno());
    echo "Connessione OK!";
    echo "<BR>";
    //Acquisizione valori utente dal form
    $DescrizioneX = $_POST['Descrizione'];
    //Costruzione stringa di query
    $query = "SELECT Videogioco.Titolo, Videogioco.DescBreve
      FROM Argomento, Videogioco
      WHERE (Argomento.CodArg = Videogioco.CodArg1)
      AND (Argomento.Descrizione = '$DescrizioneX')
      ORDER BY Titolo ASC;";
```

```

echo $query;
echo "<BR>";

//Eseguiamo la query
$resultato = mysqli_query($conn, $query) OR
die ("Query NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));

echo "Query OK!";

$righe = mysqli_num_rows($resultato);
echo "<BR>";

//Verifica se ci sono record nella query
if($righe > 0)
{
// Intestazione della tabella che conterrà lista dei videogiochi
echo "
<TABLE align='center' border='1' cellpadding='2' cellspacing='1' bgcolor='wheat' width='650'>
  <TR height='30' bgcolor = 'firebrick' >
    <TD colspan='2' >
      <DIV align='center'>
        <FONT face= 'Verdana' size = '4' color='white' >
          <B>ELENCO VIDEOGIOCHI</B>
        </FONT>
      </DIV>
    </TD>
  </TR>
";

//Riga di intestazione della tabella che conterrà lista dei dipendenti
echo "
  <TR height='30' >
    <TD align='center' width='20%'>
      <FONT face= 'Verdana' size = '3'><B>Titolo</B></FONT>
    </TD>
    <TD align='center' width='70%'>
      <FONT face= 'Verdana' size = '3'><B>Descrizione breve</B></FONT>
    </TD>
  </TR>
";

//Formattazione dei dati estratti dal result set */
while ($riga = mysqli_fetch_array($resultato, MYSQLI_BOTH))
{
// Estraggo tutti i campi presenti nella riga che è relativa a ciascun videogioco
$titolo=$riga[0];
echo "Titolo = " . $titolo . "<BR>";
$descbreve=$riga[1];
echo "Descrizione Breve = " . $descbreve . "<BR>";

//Riga ennesima della tabella che conterrà lista dei videogiochi
echo "
  <TR height='30' >
    <TD align='left' width='20%'>
      <FONT face= 'Verdana' size = '3'>$titolo</FONT>
    </TD>
    <TD align='left' width='70%'>
      <FONT face= 'Verdana' size = '3'>$descbreve</FONT>
    </TD>
  </TR>
";
}

```

```
    echo "</TABLE>";
  }
else
  {
    echo "<BR><B>Non sono presenti al momento nel nostro CATALOGO videogiochi relativi
alla descrizione dell'argomento scelto!</B>";
  }

//Chiusura connessione al database
mysqli_close($conn) OR
die ("Chiusura connessione NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));
echo "Chiusura OK!";

echo "<P align = 'center'><A href='query_a.htm'>Torna dietro</A></P>";
?>

</BODY>
</HTML>
```

LAYOUT output finale

ELENCO VIDEOGIOCHI	
Titolo	Descrizione breve
Ordinamento	Applicazione degli algoritmi di ordinamento
Ricerca	Applicazione degli algoritmi di ricerca

[Torna dietro](#)

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.

n.b. **SCELTA IMPLEMENTATIVA PER ESEGUIRE MENO JOIN:**

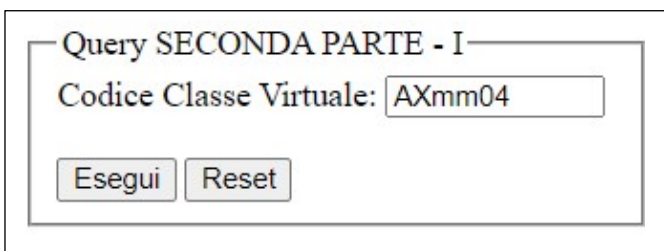
la classe virtuale è identificata dal codice presente sulla tabella **Silscrive**

```
SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
FROM Silscrive, Studente, Completa, Attivita
WHERE (Silscrive.CodS1 = Studente.CodS) AND (Studente.CodS = Completa.CodS4)
      AND (Completa.CodAtt4 = Attivita.CodAtt)
      AND (Silscrive.CodIscrizione1 = [Cod_ClasseX])
GROUP BY Studente.Cognome, Studente.Nome
ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
```

esempio con dati

```
SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
FROM Silscrive, Studente, Completa, Attivita
WHERE (Silscrive.CodS1 = Studente.CodS) AND (Studente.CodS = Completa.CodS4)
      AND (Completa.CodAtt4 = Attivita.CodAtt)
      AND (Silscrive.CodIscrizione1 = "Axmm04")
GROUP BY Studente.Cognome, Studente.Nome
ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
```

Layout FORM HTML: [query_seconda_parte_I.htm](#)



Codice HTML

```
<HTML>
  <HEAD>
    <TITLE>ESAME DI STATO 2022/2023 - SECONDA PARTE - Query I</TITLE>
  </HEAD>
  <BODY>
    <FORM name="form" action="query_seconda_parte_I.php" method="POST" target="_SELF"
      <FIELDSET>
        <LEGEND>Query SECONDA PARTE - I</LEGEND>
        <LABEL>Codice Classe Virtuale:</LABEL>
        <INPUT TYPE="text" name="CodClasse" size="10" maxlength="10"
          value="AXmm04" required ><BR><BR>
        <INPUT TYPE="submit" value="Esegui">
        <INPUT TYPE="reset" value="Reset">
      </FIELDSET>
    </FORM>
  </BODY>
</HTML>
```

SCRIPT PHP-MySQL: query_seconda_parte_1.php

```
<HTML>
  <HEAD>
    <TITLE>ESAME DI STATO 2022/2023 - SECONDA PARTE - Query I</TITLE>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </HEAD>

  <BODY>

  <?php
  // Acquisizione parametri per connessione
  include ("include\parameter.php");

  //Apertura connessione al database
  $conn = mysqli_connect($host, $utente, $pswd, $db) OR
    die("Connessione NOT OK! ".mysqli_connect_error()." ".mysqli_connect_errno());
  echo "Connessione OK!";
  echo "<BR>";

  //Acquisizione valori utente dal form
  $CodClasseX = $_POST['CodClasse'];

  //Costruzione stringa di query
  $query = "SELECT Studente.Cognome, Studente.Nome, SUM(Attivita.Monete) AS TotaleMonete
    FROM SiIscrive, Studente, Completa, Attivita
    WHERE (SiIscrive.CodS1 = Studente.CodS) AND (Studente.CodS = Completa.CodS4)
      AND (Completa.CodAtt4 = Attivita.CodAtt)
      AND (SiIscrive.CodIscrizione1 = '$CodClasseX')
    GROUP BY Studente.Cognome, Studente.Nome
    ORDER BY TotaleMonete DESC, Studente.Cognome ASC, Studente.Nome ASC;
  ";
  echo $query;
  echo "<BR>";

  //Eseguiamo la query
  $risultato = mysqli_query($conn, $query) OR
    die ("Query NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));
  echo "Query OK!";

  $righe = mysqli_num_rows($risultato);
  //echo "Numero righe ottenute = " . $righe;
  echo "<BR>";

  //Verifica se ci sono record nella query
  if($righe > 0)
  {
    // Intestazione della tabella che conterrà la classifica degli studenti
    echo "
    <TABLE align='center' border='1' cellpadding='2' cellspacing='1' bgcolor='wheat' width='650'>
      <TR height='30' bgcolor = 'firebrick' >
        <TD colspan='3' >
          <DIV align='center'>
            <FONT face= 'Verdana' size = '4' color='white' >
              <B>CLASSIFICA GENERALE STUDENTI Classe Virtuale $CodClasseX</B>
            </FONT>
          </DIV>
        </TD>
      </TR>
    ";
  }
```

```

//Prima riga di intestazione della tabella che conterrà la classifica degli studenti
echo "<TR height='30' >
    <TD align='center' width='35%'>
        <FONT face= 'Verdana' size = '3'><B>Cognome</B></FONT>
    </TD>
    <TD align='center' width='35%'>
        <FONT face= 'Verdana' size = '3'><B>Nome</B></FONT>
    </TD>
    <TD align='center' width='20%'>
        <FONT face= 'Verdana' size = '3'><B>Totale Monete</B></FONT>
    </TD>
</TR>
";

//Formattazione dei dati estratti dal result set */
while ($riga = mysqli_fetch_array($risultato, MYSQLI_BOTH))
{
    // Estraggo tutti i campi presenti nella riga
    $cognome=$riga[0];
    //echo "Cognome = " . $cognome . "<BR>";
    $nome=$riga[1];
    //echo "Nome = " . $nome . "<BR>";
    $tot_monete=$riga[2];
    //echo "Totale Monete = " . $tot_monete . "<BR>";

    //Riga ennesima della tabella che conterrà la classifica degli studenti
    echo "<TR height='30' >
        <TD align='center' width='35%'>
            <FONT face= 'Verdana' size = '3'><B>$cognome</B></FONT>
        </TD>
        <TD align='center' width='35%'>
            <FONT face= 'Verdana' size = '3'><B>$nome</B></FONT>
        </TD>
        <TD align='center' width='20%'>
            <FONT face= 'Verdana' size = '3'><B>$tot_monete</B></FONT>
        </TD>
    </TR>
";
}

echo "</TABLE>";
}
else
{
    echo "<BR><B>Non sono presenti al momento alunni con guadagno di monete nei
videogiochi assegnati alla classe virtuale $CodClasseX!</B>";
}

//Chiusura connessione al database
mysqli_close($conn) OR
die ("Chiusura connessione NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));
echo "Chiusura OK!";

echo "<P align = 'center'><A href='query_seconda_parte_I.htm'>Torna dietro</A></P>";
?>

</BODY>
</HTML>

```

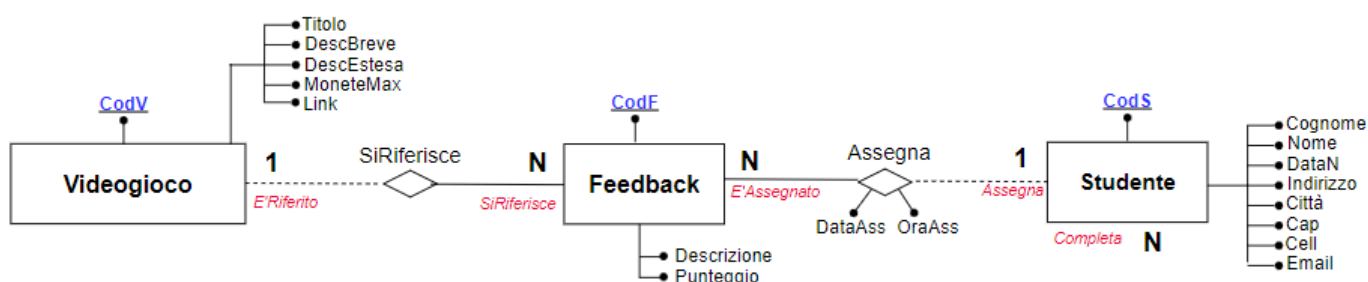
CLASSIFICA GENERALE STUDENTI Classe Virtuale AXmm04

Cognome	Nome	Totale Monete
Rossi	Franco	23
Bianchi	Emilio	19

[Torna dietro](#)

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.

Si propone la seguente integrazione al diagramma ER presentato in precedenza aggiungendo la nuova entità "Feedback" collegandola con le due associazioni binarie "Assegna" e "SiRiferisce" rispettivamente alle entità "Studente" e "Videogioco".



Ovviamente sarà possibile assegnare un feedback solo a quei videogiochi assegnati alla classe virtuale cui lo studente si è in precedenza regolarmente iscritto e dei quali ha completato tutte le attività previste (fino al raggiungimento del numero di monete previsto nell'attributo Videogioco.MoneteMax)

Si osservi che gli schemi delle relazioni "Studente" e "Videogioco" non subiscono alcuna modifica dall'inserimento di questa nuova relazione.

Il mapping logico-relazionale verrà arricchito dall'introduzione dei seguenti "oggetti":

Feedback (CodF, Descrizione, Punteggio, DataAss, OraAss, CodS5, CodV5)

con l'attributo "CodS5" della relazione "Feedback" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodS" della relazione "Studente"

con l'attributo "CodV5" della relazione "Feedback" che risulta essere chiave esterna o foreign key (FK) sull'attributo "CodV" della relazione "Videogioco"

$VR_{CodS5}(\text{Feedback}) \subseteq VR_{CodS}(\text{Studente})$

Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione INVERSA "E'Assegnato"

$VR_{CodV5}(\text{Feedback}) \subseteq VR_{CodV}(\text{Videogioco})$

Vincolo Referenziale (VR) causato dalla TOTALITA' dell'associazione DIRETTA "SiRiferisce"

$V4 : (\text{Feedback.Punteggio BETWEEN 1 AND 5}) \implies V4(\text{Feedback}) : (\text{Punteggio BETWEEN 1 AND 5})$

Vincolo intrarelazionale o interno su SINGOLA ennupla sul dominio di UN ATTRIBUTO

Utilizzando il linguaggio SQL dovrà essere aggiunta nel database la nuova tabella Feedback

CREATE TABLE Feedback

```
(
CodF          INT (10) NOT NULL AUTO_INCREMENT,
Descrizione   VARCHAR (160) NOT NULL,
Punteggio     INT (1) NOT NULL,
DataAss       DATE NOT NULL,
OraAss        TIME NOT NULL,
CodS5         VARCHAR (10) NOT NULL,
CodV5         VARCHAR (10) NOT NULL,
PRIMARY KEY (CodF),
```


FOREIGN KEY (CodS5) REFERENCES Studente (CodS),
FOREIGN KEY (CodV5) REFERENCES Videogioco (CodV),
CHECK (Punteggio BETWEEN 1 AND 5)
) Engine InnoDB DEFAULT CHARSET=utf8 COLLATE = utf8_general_ci;

#Vincolo V4

Volendo far inserire tramite form HTML un feedback ad uno studente (identificato dal suo codice) su un determinato videogioco (anch'esso identificato ramite codice) utilizzando uno script in PHP che si interfacci con il database server MySQL potremmo creare:

Layout FILE HTML: **query_seconda_parte_II.htm**



Query SECONDA PARTE - II

Codice Studente:

Codice Videogioco:

Punteggio:

Feedback:

Codice HTML

```
<HTML>
<HEAD>
  <TITLE>ESAME DI STATO 2022/2023 - SECONDA PARTE - Query II</TITLE>
</HEAD>
<BODY>
  <FORM name="form" action="query_seconda_parte_II.php" method="POST" target="_SELF">
    <FIELDSET>
      <LEGEND>Query SECONDA PARTE - II</LEGEND>
      <LABEL>Codice Studente:</LABEL>
      <INPUT TYPE="text" name="CodStudente" size="10"
        maxlength="10" value="S-001" required ><BR><BR>
      <LABEL>Codice Videogioco:</LABEL>
      <INPUT TYPE="text" name="CodVideogioco" size="10"
        maxlength="10" value="V-001" required ><BR><BR>
      <LABEL>Punteggio:</LABEL>
      <INPUT TYPE="number" name="Punteggio" min = "1" max = "5"
        step = "1" value = "1" required ><BR><BR>
      <LABEL>Feedback:</LABEL>
      <TEXTAREA name="DescFeedback" rows="4" cols="40" maxlength="160" required >
      </TEXTAREA><BR><BR>
      <INPUT TYPE="submit" value="Esegui">
      <INPUT TYPE="reset" value="Reset">
    </FIELDSET>
  </FORM>
</BODY>
</HTML>
```

SCRIPT PHP-MySQL: query_seconda_parte_II.php

```
<HTML>
  <HEAD>
    <TITLE>ESAME DI STATO 2022/2023 - SECONDA PARTE - Query II</TITLE>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </HEAD>

  <BODY>

<?php
// Acquisizione parametri per connessione
include ("include\parameter.php");

//Apertura connessione al database
$conn = mysqli_connect($host, $utente, $pswd, $db) OR
    die("Connessione NOT OK! ".mysqli_connect_error()." ".mysqli_connect_errno());
//echo "Connessione OK!";
echo "<BR>";

//Acquisizione valori utente dal form
$CodStudenteX = $_POST['CodStudente'];
$CodVideogiocoX = $_POST['CodVideogioco'];
$PunteggioX = $_POST['Punteggio'];
$DescFeedbackX = $_POST['DescFeedback'];

/*
echo "Codice studente = $CodStudenteX <BR>";
echo "Codice videogioco = $CodVideogiocoX <BR>";
echo "Punteggio = $PunteggioX <BR>";
echo "Descrizione feedback = $DescFeedbackX <BR>";
*/

$DataAssX = date ('Y-m-d', time());
//echo "Data Assegnazione = $DataAssX <BR>";
$OraAssX = date ('H:i:s', time());
//echo "Ora Assegnazione = $OraAssX <BR>";

//N.B.
// 1) Ovviamente l'operazione di INSERT del feedback andrebbe preceduta da un controllo
// che verifichi il completamento da parte dello studente per quel videogioco di tutte
// le attività ad esso correlate
// 2) Questo script prevede il controllo se lo studente per quel videogioco abbia già
// assegnato in precedenza un feedback permettendogli l'aggiornamento.
// Semplicemente lo studente può inserire una volta il feedback per un certo
// videogioco poi può solo modificarne descrizione e punteggio.
//Costruzione stringa di query
$query = "SELECT CodF
        FROM Feedback
        WHERE (CodS5 = '$CodStudenteX') AND (CodV5 = '$CodVideogiocoX');
        ";
echo $query;
echo "<BR>";

//Eseguiamo la query
$resultato = mysqli_query($conn, $query) OR
    die ("Query NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));
//echo "Query OK!";
echo "<BR>";
```

```

$righe = mysqli_num_rows($risultato);
//echo "Numero righe trovate = " . $righe;
echo "<BR>";

// valuto se lo studente ha o non ha già espresso il suo feedback per quel videogioco
if ($righe == 0)
{
    echo "Lo studente $CodStudenteX NON MAI ESPRESSO FEEDBACK per il videogioco
$CodVideogiocoX <BR>";

    //Costruzione stringa di query
    $query = "INSERT INTO Feedback
                (CodF, Descrizione, Punteggio, DataAss, OraAss, CodS5, CodV5)
                VALUES (NULL, '$DescFeedbackX', $PunteggioX, '$DataAssX', '$OraAssX',
                '$CodStudenteX', '$CodVideogiocoX')";

    ";
    echo $query;
    echo "<BR>";

    //Eseguiamo la query
    $risultato = mysqli_query($conn, $query) OR
                die ("Query NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));
    //echo "INSERT OK!";
    echo "<BR>";

    $righe = mysqli_affected_rows($conn);
    //echo "Numero righe coinvolte = " . $righe;
    echo "<BR>";

    //Verifica se l'operazione di insert è andata a buon fine
    if($righe == 1)
    {
        echo "<BR><P align = 'center'><B>Operazione di inserimento Feedback avvenuta con
successo!</B><BR><BR></P>";
    }
    else
    {
        echo "<BR><P align = 'center'><B>ERRORE nell'operazione di inserimento!</B></P>";
    }
}
else
{
    echo "Lo studente $CodStudenteX HA GIA' ESPRESSO FEEDBACK per il videogioco
$CodVideogiocoX <BR>";

    $riga = mysqli_fetch_array($risultato, MYSQLI_ASSOC);
    // Estraggo il codice del feedback presente nella riga
    $CodF = $riga['CodF'];
    //echo "Codice feedback = $CodF <BR>";

    //Costruzione stringa di query
    $query = "UPDATE Feedback
                SET Descrizione = '$DescFeedbackX',
                    Punteggio = $PunteggioX,
                    DataAss = '$DataAssX',
                    OraAss = '$OraAssX'
                WHERE CodF = $CodF;

    ";
    echo $query;
    echo "<BR>";

    //Eseguiamo la query
    $risultato = mysqli_query($conn, $query) OR
                die ("Query NOT OK! ".mysqli_error($conn)." ".mysqli_errno($conn));
    //echo "UPDATE OK!";
}

```

```

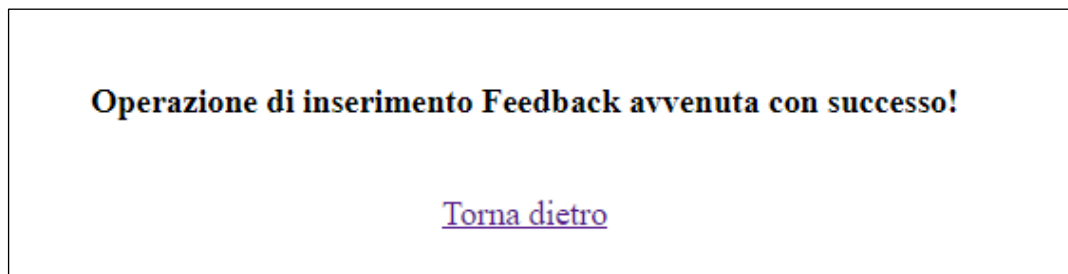
echo "<BR>";
$righe = mysqli_affected_rows($conn);
//echo "Numero righe coinvolte = " . $righe;
echo "<BR>";
//Verifica se l'operazione di update è andata a buon fine
if($righe == 1)
{
    echo "<BR><P align = 'center'><B>Operazione di aggiornamento Feedback avvenuta
con successo!</B><BR><BR></P>";
}
else
{
    echo "<BR><P align = 'center'><B>ERRORE nell'operazione di
aggiornamento!</B></P>";
}
}
//Chiusura connessione al database
mysqli_close($conn) OR
    die ("Chiusura connessione NOT OK! " . mysqli_error($conn). " " . mysqli_errno($conn));
//echo "Chiusura OK!";
echo "<P align = 'center'><A href='query_seconda_parte_II.htm'>Torna dietro</A></P>";
?>

</BODY>
</HTML>

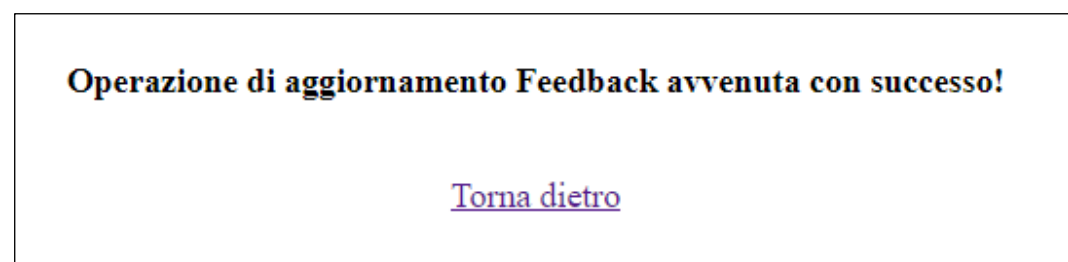
```

LAYOUT output finale

Se è la prima volta che lo studente lascia un feedback per quel video gioco allora verrà mostrato a video il seguente messaggio:



Se NON è la prima volta che lo studente lascia un feedback per quel video gioco allora verrà mostrato a video il seguente messaggio:



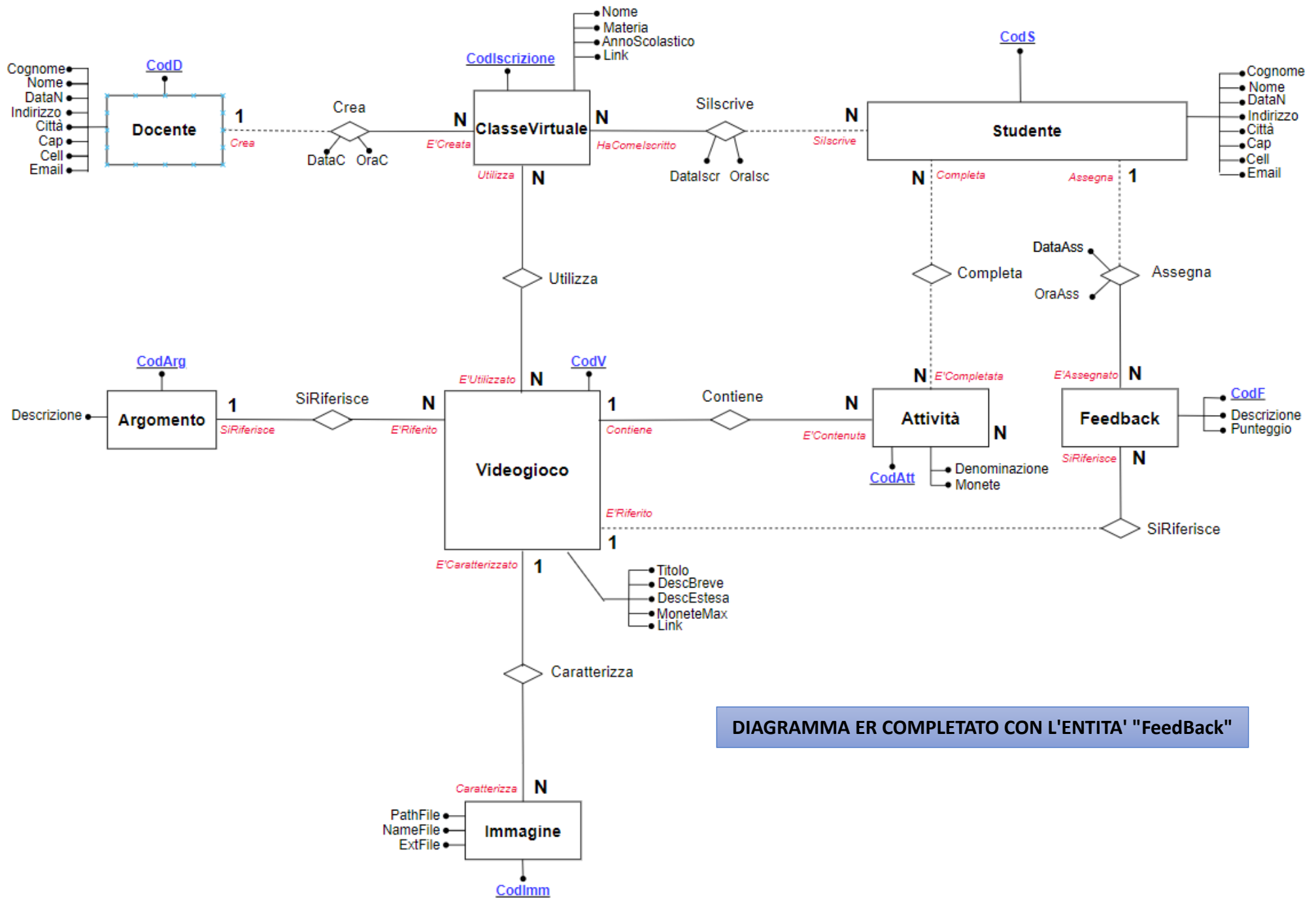


DIAGRAMMA ER COMPLETATO CON L'ENTITA' "FeedBack"

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.

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una tabella; a tale scopo si utilizzano delle funzioni speciali che restituiscono un solo valore, e come tali concorrono a creare tabelle aventi un'unica riga.

Tali **funzioni** o **operatori** di aggregazione si applicano in generale ad una colonna di una tabella all'interno di una istruzione **SELECT**. Esse sono cinque:

- la funzione **COUNT**: **può essere usata su tutti i tipi di dati** e conta il numero di elementi (righe) presenti in una tabella in un certo istante di tempo relativamente ad una o più colonne ossia restituisce la **cardinalità** della relazione da cui è stata generata (può avere più colonne come argomenti);
- la funzione **MIN**: **può essere usate su qualsiasi dato ordinabile numerico o alfanumerico**, e restituisce il valore minimo tra i valori della colonna, anche di tipo carattere, specificata come argomento della funzione;
- la funzione **MAX**: **può essere usate su qualsiasi dato ordinabile numerico o alfanumerico**, e restituisce il valore massimo tra i valori della colonna, anche di tipo carattere, specificata come argomento della funzione;
- la funzione **SUM**: **può essere usata solo su dati numerici** e restituisce la somma di tutti i valori contenuti in una colonna specificata come argomento della funzione;
- la funzione **AVG** (dall'inglese Average): **può essere usato solo su dati numerici o intervalli di tempo** e calcola la media aritmetica dei valori numerici contenuti in una determinata colonna di una tabella, con l'eventuale aggiunta dell'opzione **DISTINCT**.

Capita con una notevole frequenza di dover effettuare calcoli o statistiche su dati memorizzati, che abbiano qualche caratteristica in comune.

Il linguaggio SQL mette a disposizione, a questo scopo, un potente strumento: il **raggruppamento** (attraverso la clausola **GROUP BY** all'interno della **SELECT**)

"Raggruppare" significa "mettere insieme" tutti i dati, accomunati da qualche caratteristica, su cui vanno fatti alcuni tipi di conteggi.

Le funzioni di aggregazione che, come abbiamo visto in precedenza, permettono di ottenere informazioni riepilogative di sintesi delle ennuple di una tabella, sono in genere abbinata alle **clausole di raggruppamento** all'interno dell'istruzione **SELECT**.

La forma generale della **SELECT** diventa allora:

```
SELECT <Attr_1> [, <Attr_2>, ...<Attr_N>] [, <funz_raggr_1>,....., <funz_raggr_N>]
FROM <Tab_1> [, <Tab_2>, ...<Tab_N>]
WHERE <lista_condizioni>
GROUP BY <Attr_1> [, <Attr_2>, ...<Attr_N>] [, <funz_raggr_1>,....., <funz_raggr_N>]
[HAVING <Condizione_Gruppo>]
```

Per esempio si determina con una query di raggruppamento l'individuazione dello stipendio minimo o massimo dei dipendenti di un determinato reparto (codice); oppure la somma degli stipendi dei dipendenti con un determinato livello e via dicendo

Da quanto detto, possiamo desumere che se viene specificata una clausola **GROUP BY**, allora nella clausola **SELECT** dovrà essere presente:

- o un campo specificato nella clausola **GROUP BY**;
- oppure una **funzione di aggregazione**.

Questo perché quando il motore aggrega le righe deve sapere come comportarsi per ogni campo da restituire.

Per raggruppare i dipendenti in base al loro livello e conoscere lo stipendio medio per livello possiamo scrivere:

```
SELECT Livello, AVG(Stipendio)
FROM Dipendente
GROUP BY Livello;
```

Livello	AVG(Stipendio)
7	1200.000000
8	1850.000000
9	4200.000000

N.B. Nell'utilizzo nell'ambiente **MySQL** è **sintatticamente errato** lasciare uno spazio tra il nome della funzione di aggregazione e la prima parentesi tonda aperta!

Filtraggio sul raggruppamento

A differenza di WHERE, che agisce a livello di singola riga, la parola chiave opzionale **HAVING** permette di effettuare un filtraggio sul raggruppamento.

Questa clausola si inserisce subito dopo la **GROUP BY**.

Il criterio di filtraggio può contenere una qualsiasi funzione di raggruppamento ma anche un altro eventuale predicato logico:

Per raggruppare i dipendenti in base al loro livello e conoscere lo stipendio medio a partire dall'8° livello, possiamo scrivere:

```
SELECT Livello, AVG(Stipendio)
FROM Dipendente
GROUP BY Livello
HAVING Livello >= 8 ;
```

```
+-----+-----+
| Livello | AVG(Stipendio) |
+-----+-----+
|      8 | 1850.000000 |
|      9 | 4200.000000 |
+-----+-----+
```

Nota bene

Nell'istruzione **SELECT**, la differenza principale tra la clausola opzionale **WHERE** (detta "taglio orizzontale") e la clausola opzionale **HAVING** sui raggruppamenti (detto "taglio orizzontale sui gruppi") è che la clausola **WHERE** viene utilizzata per filtrare i record (ossia le singole ennuple) **PRIMA** che si verifichi un raggruppamento o un'aggregazione, mentre **HAVING** si utilizza per filtrare i record **DOPO** un raggruppamento o che si è verificata un'aggregazione.

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

Una **forma normale** è una proprietà di uno *schema relazionale* che ne garantisce la qualità misurata in assenza di determinati difetti.

Con **normalizzazione** si intende il procedimento che serve a trasformare uno schema che presenta **anomalie** (*schema non normalizzato*) in uno **equivalente** (con lo stesso contenuto informativo) in cui tali anomalie sono state eliminate (*schema normalizzato*).

Quando uno schema relazionale non è normalizzato può comportare **comportamenti non desiderati** che possono compromettere le operazioni di congruenza durante le operazioni di:

- **inserimento** dei dati;
- **aggiornamento** dei dati;
- **cancellazione** dei dati;

Nel nostro esempio

a) anomalie di inserimento:

- non è possibile inserire un nuovo *Progetto* senza inserire i dati relativi al relativo *Tutor*
- non è possibile inserire un nuovo *Tutor* senza inserire i dati relativi ad un *Progetto*

b) anomalie di aggiornamento:

- per modificare il *numero di telefono* di un *Tutor* occorre modificare tutte le ennuple in cui compare
- per modificare il *tipo* di un *Progetto* occorre modificare tutte le ennuple in cui compare

Se le modifiche fossero parziali si lascerebbe la base dei dati in uno stato detto **inconsistente**

c) anomalie di cancellazione:

- cancellando la ennupla *ID = 1* si perdono le informazioni relative al *Progetto* "Pensiero Computazionale"

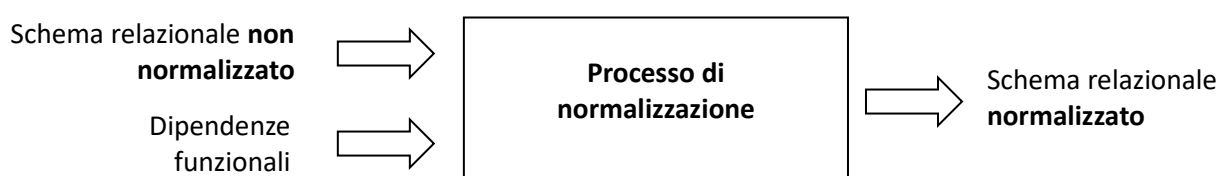
Queste anomalie si verificano perché abbiamo rappresentato informazioni eterogenee tra loro con un'unica relazione.

Il **processo di normalizzazione** elimina tali **anomalie** effettuando una serie di trasformazioni successive delle relazioni di partenza di uno schema relazionale ottenendo altre relazioni che a seconda del tipo di trasformazione applicata, possono rispondere a diversi livelli di "bontà" dette **forme normali**.

Esistono molte forme normali per uno schema relazionale:

- prima forma normale o **1FN**;
- seconda forma normale o **2FN**;
- terza forma normale o **3FN** con definizione alternativa di di Boyce-Codd o **BCFN**;
- quarta forma normale o **4FN**;
- quinta forma normale o **5FN**.

Per i nostri scopi è sufficiente applicare il processo di normalizzazione per ottenere uno schema relazionale in **terza forma normale o 3FN** e vedremo come ottenerlo attraverso lo studio delle **dipendenze funzionali**.



Diremo che una relazione **R** è in **prima forma normale** o **1FN** quando rispetta i **requisiti fondamentali** del modello relazionale che sono:

- i **valori di un attributo** (di una colonna) sono dello stesso tipo ovvero appartengono allo stesso dominio;
- i **valori di una ennupla** (di una riga) sono diversi da quelli delle altre ennuple ovvero non possono esistere due ennuple uguali;
- l'**ordine delle ennuple** è irrilevante;
- gli **attributi** sono di tipo **elementare** ossia:
 - non possono *essere composti* da gruppi di attributi ripetuti (no attributi composti o aggregati);
 - non possono *prevedere* una lista variabile di valori (no attributi multipli).

La nostra relazione "Progetti" non è **1FN** perché l'attributo **Tutor** è composto in realtà al suo **Nome** e dal suo **Cognome**;

mentre rispetta tutte le altre condizioni espresse nella definizione di **1FN**.

L'attributo **ID** svolge la funzione di chiave primaria o primary key

Passaggio in 1FN

ID	Titolo	Budget	Tipo	DataInizio	DataFine	Tutor Cognome	Tutor Nome	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

Data una relazione **R** ed un insieme $X = \{ X_1, X_2, \dots, X_N \}$ di **R** si dice che un attributo **Y** di **R** **dipende funzionalmente da X** e si scrive:

$$X_1, X_2, \dots, X_N \rightarrow Y$$

se e solo se i valori degli attributi di **X** determinano univocamente il valore dell'attributo **Y** per ogni istanza della relazione **R**.

Si dice anche che l'insieme **X determina Y**.

Nella nostra relazione "Progetti" accade che:

- 1) **ID** → **Titolo**: il Titolo dipende funzionalmente da quel determinato Progetto
- 2) **ID** → **Budget**: Il Budget dipende funzionalmente da quel determinato Progetto
- 3) **ID** → **DataInizio**: la DataInizio dipende funzionalmente da quel determinato Progetto
- 4) **ID** → **DataFine**: la DataFine dipende funzionalmente da quel determinato Progetto

Riassumendo in breve possiamo dire che ID → Titolo, Budget, DataInizio, DataFine

- 1) **TutorCognome, TutorNome** → **TelTutor**: Il telefono dal tutor dipende funzionalmente da quel determinato Tutor (con nome e cognome specifici)

Riassumendo in breve possiamo dire che TutorCognome, TutorNome → TelTutor

OSS: la coppia di attributi **TutorCognome** e **TutorNome** non sono in grado di distinguere ciascuna ennupla della relazione "Progetti" in quanto in caso di omonimia non potremmo distinguerli tra loro

Quindi per porre la relazione in 3FN occorre distinguere le informazioni relative al progetto rispetto a quelle relative ai Tutor creando le seguenti 2 relazioni:

Passaggio in 3FN: Creazione Relazione "Tutor"

ID_Tutor	Cognome	Nome	Telefono
T-001	Rossi	Mario	345678910
T-002	Bianchi	Carlo	333444555

Passaggio in 3FN: Trasformazione Relazione "Progetto"

ID_Prg	Titolo	Budget	Tipo	DataInizio	DataFine	ID_Tutor
1	Pensiero computazionale	40.000	PON	20/02/2023	NULL	T-001
2	Robotica educativa	13.000	PCTO	10/11/2022	30/03/2023	T-002
3	Tinkering	25.000	PCTO	14/10/2022	20/02/2023	T-002
4	Realtà virtuale	30.000	PCTO	16/02/2023	30/05/2023	T-001

Sebbene esistano la **4FN** e la **5FN** non vengono considerate in questo contesto.

La forma normale obbligatoria è la prima ossia la 1FN mentre le altre servono per separare meglio concetti eterogenei racchiudendoli in relazioni che al loro interno siano il più possibile omogenee.

Come si è potuto vedere passando dalla **1FN** alle altre forme normali è VERO che si **eliminano** le anomalie ma si **introducono** INEVITABILI ridondanze (ripetizioni) nella base di dati.