Premessa

Da quest'anno, secondo le linee guida del MIUR per la riforma degli istituti tecnici, la tipologia della seconda prova di Informatica prevede l'introduzione di una seconda parte relativa a quesiti.
Il MIUR, soltanto il 16 aprile 2015 (quindi appena due mesi prima dell'inizio dell'esame di stato), ha pubblicato un solo esempio di seconda prova di Informatica.
Tale esempio di seconda prova evidenzia più di una problematica e più precisamente:
1) la prima parte è strutturata allo stesso modo delle prove degli anni precedenti e cioè progettazione concettuale, logica, definizione della struttura delle tabelle, query (piuttosto complesse nella proposta del MIUR) e parte web e anche il livello di complessità non è inferiore alle prove degli anni precedenti
2) il tempo a disposizione è sempre di sei ore ma in tale tempo devono essere risolti anche i quesiti
3) negli anni precedenti svolgere completamente e correttamente un problema analogo a quello proposto nella prima parte del testo del MIUR avrebbe consentito agli studenti di conseguire un punteggio di 15/15 mentre ora, per arrivare al punteggio massimo, è necessario rispondere correttamente e nello stesso tempo anche ai quesiti
4) nella seconda parte del testo del MIUR si richiede di rispondere ad almeno due quesiti; questo cosa significa? Rispondere correttamente a due quesiti implica una valutazione solo sufficiente e per raggiungere il punteggio massimo lo studente deve rispondere a tutti i quesiti? Oppure significa che rispondere correttamente a due quesiti corrispondere al conseguimento del punteggio massimo? E ancora, se la richiesta è di rispondere a due quesiti per avere il massimo, come considerare uno studente che copia in bella la risposta a tutti i quesiti? Si considerano solo quelli corretti?

Ritengo che dovrà essere il commissario esterno a dare delle indicazioni chiare agli studenti dopo avere preso visione del testo della seconda prova di Informatica.

Commento al tema proposto

Il testo è chiaro e non emergono ambiguità e/o contraddizioni, come è avvenuto in passato in alcune occasioni nelle tracce proposte agli esami di Stato. I quesiti sono coerenti con i programmi che generalmente vengono svolti nel 5° anno e gli studenti che hanno seguito con regolarità le lezioni hanno tutti i mezzi per affrontare positivamente la prova. Lo schema concettuale e lo schema logico, pur non essendo particolarmente complessi, richiedono tempo, approfondimento e riflessione. Per quel che riguarda le query, che non sono semplici e coinvolgono più tabelle, non è facile per gli studenti proporre soluzioni "sulla carta", che non possono testare sul computer, esenti da errori. Molto impegnativa in termini di tempo risulta l'elaborazione dei quesiti 6 e7. Infine i quesiti relativi alla seconda parte (la novità dell'esame di Stato 2015) risultano alla portata degli studenti preparati e si prestano sia ad una trattazione sintetica che ad un'elaborazione articolata.
La prova è valida e ben strutturata ma risulta eccessivamente lunga e laboriosa. Nessun problema per un'esercitazione. Ma se si trattasse della prova di esame, anche per i candidati più preparati sarebbe difficile svolgerla in modo completo in sole sei ore.

Soluzione prima parte

1 un'analisi della realtà di riferimento individuando le possibili soluzioni e scelga quella che a suo motivato giudizio è la più idonea a rispondere alle specifiche indicate

La soluzione da adottare è indicata chiaramente nella traccia quando si precisa che "*tali informazioni vengono inserite direttamente dai caseifici a fine giornata, mediante accesso ad un'area riservata dell'interfaccia Web del sito del consorzio*" e consiste in una classica web application cui ciascun caseificio accede tramite credenziali riservate dal sito web del consorzio. Quindi occorre che ciascun caseificio disponga di un pc (o una rete di pc a seconda delle dimensioni del caseificio) collegato alla rete Internet.

Riguardo al software si ipotizza di utilizzare

- MySql come Database
- Apache come Web Server
- Php come linguaggio lato Server
- HTML, CSS, Java Script per implementare le pagine Web

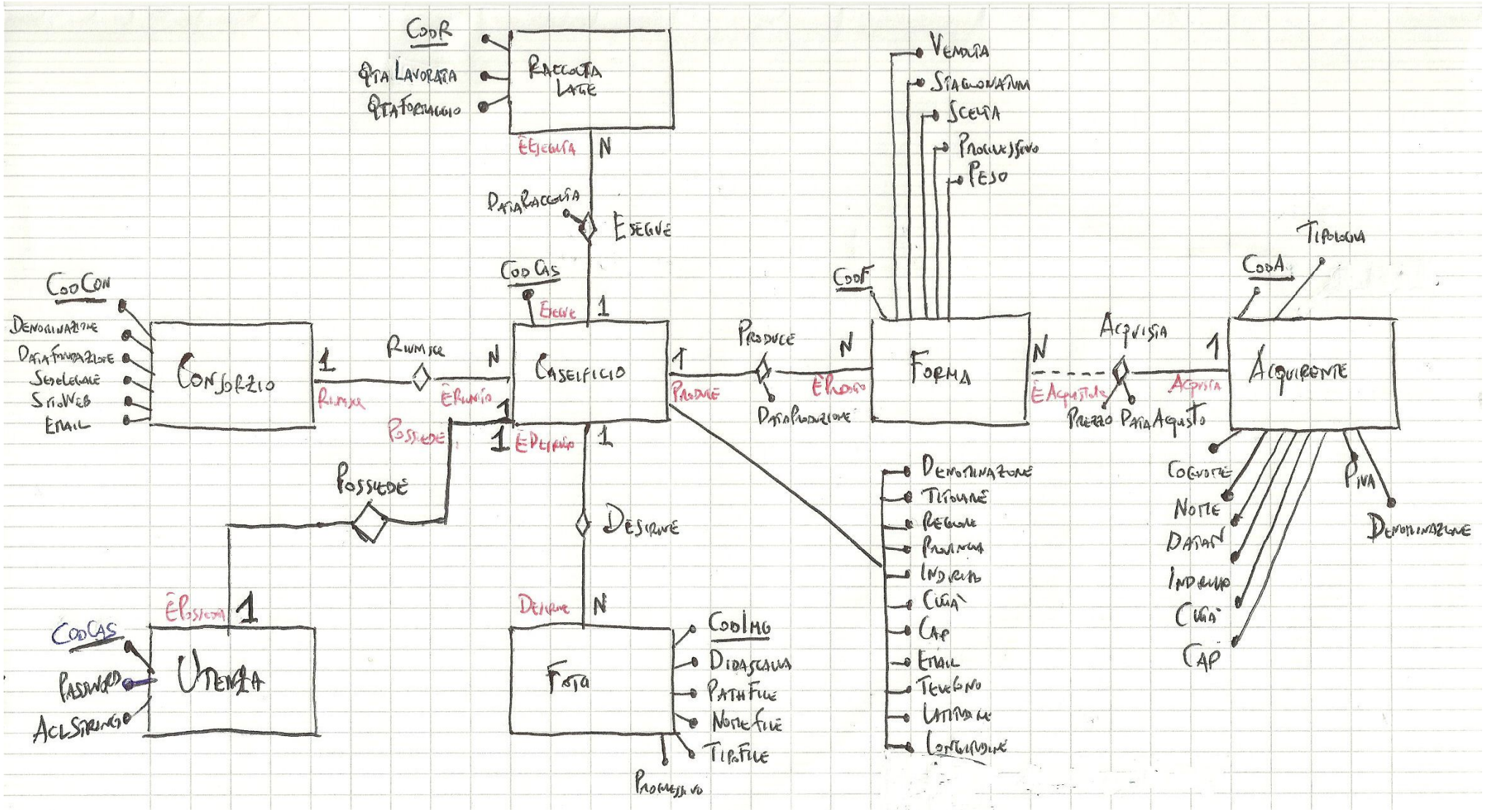
Si ipotizza inoltre che il consorzio opera su un territorio che interessa geograficamente più di una provincia

Per ogni caseificio verranno memorizzati i dati identificativi (denominazione, regione, provincia, indirizzo, ecc) ivi comprese le coordinate in termini di latitudine e longitudine che permetteranno, nell'ambito di un sito web di presentazione, la sua georeferenziazione in un ambiente quale ad esempio Google-map.

Per quanto riguarda le foto che permetteranno sullo stesso sito il tour virtuale dell'insediamento produttivo, si ipotizza di associare ad ogni foto un numero progressivo che permette di definire la sequenza ordinata del tour stesso.
Per una scelta di praticità, tutti i file delle foto verranno memorizzati in una specifica cartella, mentre nella tabella relativa alle foto verrà registrato il *pathname* dei file immagine (es. "Consorzio\Img\CaseificioCAS1.jpg") piuttosto che le immagini stesse in campi di tipo blog

Dal momento che vi possono essere clienti abituali è ragionevole mantenerne i dati identificativi archiviati in una tabella dedicata

2. lo schema concettuale della base di dati *(nel nostro caso DIAGRAMMA ER)*



PREMESSA teorica

*La **progettazione concettuale** ha lo scopo di costruire e definire una rappresentazione corretta e completa della realtà di interesse concettuale in modo astratto ed indipendente dal DBMS che si intenderà utilizzare.*

L'input di tale fase è il documento delle specifiche formali (nel nostro caso il testo della traccia di esame)

L'output di tale fase è uno schema concettuale ossia una rappresentazione astratta (indipendente dal DBMS) ed il più possibile formale della realtà (ad esempio il diagramma E/R).

2.1 Regole di lettura

Un consorzio deve riunire uno o più caseifici e viceversa un caseificio deve essere riunito da un ed un sol consorzio

Un caseificio deve eseguire una o più raccolta del latte e viceversa una raccolta del latte deve essere eseguita da un ed un sol caseificio

Una foto deve descrivere un ed un solo caseificio e viceversa un caseificio deve essere descritto da una o più foto

Un caseificio deve produrre una o più forme di formaggio e viceversa una forma di formaggio deve essere prodotta da un ed un sol caseificio

Un caseificio deve possedere una ed una sola utenza e viceversa una utenza deve essere posseduta da un ed un sol caseificio

Un acquirente deve acquistare una o più forme di formaggio e viceversa una forma di formaggio può essere acquistata da nessun o un acquirente

2.2 Vincoli di integrità: *è un'asserzione ovvero un predicato che deve essere soddisfatto da tutte le istanze di una categoria (nel nostro caso le entità)*

Vincoli di integrità impliciti *(sono quei vincoli di integrità direttamente deducibili dal diagramma ER in quanto caratterizzati da particolari segni grafici)*

- V.d.I. impliciti di chiave primaria:

  l'attributo "CodCon" è chiave primaria (in breve PK) dell'entità "Consorzio";
  l'attributo "CodCas" è PK dell'entità "Caseificio";
  l'attributo "CodCas" è PK dell'entità "Utenza";
  l'attributo "CodR" è PK dell'entità "RaccoltaLatte";
  l'attributo "CodImg" è PK dell'entità "Foto";
  l'attributo "CodF" è PK dell'entità "Forma";
  l'attributo "CodA" è PK dell'entità "Acquirente";

- V.d.I. impliciti causati dalla totalità delle associazioni dirette e/o inverse:

  totalità associazione diretta "Riunisce" tra le entità "Consorzio" e "Caseificio";
  totalità associazione inversa "E'Riunito" tra le entità "Caseificio" e "Consorzio";
  totalità associazione diretta "Possiede" tra le entità "Caseificio" e "Utenza";
  totalità associazione inversa "E'Posseduta" tra le entità "Utenza" e "Caseificio";
  totalità associazione diretta "Esegue" tra le entità "Caseificio" e "RaccoltaLatte";
  totalità associazione inversa "E'Eseguita" tra le entità "RaccoltaLatte" e "Caseificio";
  totalità associazione diretta "Descrive" tra le entità "Foto" e "Caseificio";
  totalità associazione inversa "E'Descritto" tra le entità "Caseificio" e "Foto";
  totalità associazione diretta "Produce" tra le entità "Caseificio" e "Forma";
  totalità associazione inversa "E'Prodotta" tra le entità "Forma" e "Caseificio";
  totalità associazione diretta "Acquista" tra le entità "Acquirente" e "Forma";

Vincoli di integrità espliciti *(sono quei vincoli di integrità non direttamente deducibili dal diagramma ed esprimibili attraverso opportune dichiarazioni)*

V1: (Acquirente.Tipologia IN ("PR", "GR", "GD"))

dove "PR" è per "Privato", "GR" è per "Grossista" e "GD" è per "Grande Distribuzione"

V2: (Forma.Scelta IN (1, 2))

dove 1 è per prima scelta e 2 per seconda scelta

V3: (Forma.Stagionatura IN (12, 24, 30, 36))

V4: (Foto.TipoFile IN ("JPG", "GIF", "BMP"))

V5: (Forma.Venduta IN ("SI", "NO"))

3. lo schema logico della base dati *(nel nostro caso schema logico-relazionale)*

PREMESSA teorica

*La **progettazione logica** ha lo scopo di trasformare lo schema concettuale (ancora astratto e indipendente da un DBMS) in uno schema logico ovvero in una rappresentazione efficiente rispetto alle strutture di un DBMS relativamente ad un ben definito modello dati (un esempio è una descrizione tramite tabelle del modello relazionale).*

L'input di tale fase è il diagramma ER ottenuto alla fine della precedente fase di progettazione concettuale.

L'output di tale fase è uno schema logico riassumibile con relazioni rappresentate da tabelle logiche (schema logico relazionale)

a) mapping relazionale dell'associazione binaria "Riunisce" tra le entità "Consorzio" e "Caseificio" di molteplicità 1:N

Consorzio (CodCon, Denominazione, DataFondazione, SedeLegale, SitoWeb, Email)

Caseificio (CodCas, Denominazione, Titolare, Regione, Provincia, Indirizzo, Città, Cap, Email, Telefono, Latitudine, Longitudine, CodCon1)

con l'attributo "CodCas" della relazione "Caseificio" che risulta essere chiave esterna (ossia foreign key o in breve FK) sull'attributo "CodCas" della relazione "Utenza"

con l'attributo "CodCon1" della relazione "Caseificio" che risulta essere chiave esterna (ossia foreign key o in breve FK) sull'attributo "CodCon" della relazione "Consorzio"

VR_{CodCon} (Consorzio) $\subseteq$ $VR_{CodCon1}$ (Caseificio) Vincolo referenziale (in breve VR) causato dalla totalità dell'associazione diretta "Riunisce"

$VR_{CodCon1}$ (Caseificio) $\subseteq$ VR_{CodCon} (Consorzio) VR causato dalla totalità dell'associazione inversa "E'Riunito"

b) mapping relazionale dell'associazione binaria "Esegue" tra le entità "Caseificio" e "RaccoltaLatte" di molteplicità 1:N

Caseificio : relazione già mappata in precedenza

RaccoltaLatte (CodR, QtaLavorata, QtaFormaggio, DataRaccolta, CodCas1)

con l'attributo "CodCas1" della relazione "RaccoltaLatte" che risulta essere FK sull'attributo "CodCas" della relazione "Caseificio"

VR_{CodCas} (Caseificio) $\subseteq$ $VR_{CodCas1}$ (RaccoltaLatte) VR causato dalla totalità dell'associazione diretta "Esegue"

$VR_{CodCas1}$ (RaccoltaLatte) $\subseteq$ VR_{CodCas} (Caseificio) VR causato dalla totalità dell'associazione inversa "E'Eseguita"

c) mapping relazionale dell'associazione binaria "Descrive" tra le entità "Foto" e "Caseificio" di molteplicità N:1

Foto (CodImg, Didascalia, PathFile, NomeFile, TipoFile, Progressivo, CodCas2)

con l'attributo "CodCas2" della relazione "Foto" che risulta essere FK sull'attributo "CodCas" della relazione "Caseificio"

Caseificio : relazione già mappata in precedenza

$VR_{CodCas2}$ (Foto) $\subseteq$ VR_{CodCas} (Caseificio) VR causato dalla totalità dell'associazione diretta "Descrive"

VR_{CodCas} (Caseificio) $\subseteq$ $VR_{CodCas2}$ (Foto) VR causato dalla totalità dell'associazione inversa "E'Descritto"

d) mapping relazionale dell'associazione binaria "Possiede" tra le entità "Caseificio" e "Utenza" di molteplicità 1:1

Caseificio : relazione già mappata in precedenza

Utenza (CodCas, Password, ACLStringa)

VR_{CodCas} (Caseificio) $\subseteq$ VR_{CodCas} (Utenza) VR causato dalla totalità dell'associazione diretta "Possiede"

VR_{CodCas} (Utenza) $\subseteq$ VR_{CodCas} (Caseificio) VR causato dalla totalità dell'associazione inversa "E'Posseduta"

e) mapping relazionale dell'associazione binaria "Produce" tra le entità "Caseificio" e "Forma" di molteplicità 1:N

Caseificio : relazione già mappata in precedenza

Forma (CodF, Venduta, Stagionatura, Scelta, Progressivo, Peso, DataProduzione, CodCas3, Prezzo, DataAcquisto, CodA1)

con l'attributo "CodCas3" della relazione "Forma" che risulta essere FK sull'attributo "CodCas" della relazione "Caseificio"

con l'attributo "CodA1" della relazione "Forma" che risulta essere FK sull'attributo "CodA" della relazione "Acquirente"

VR_{CodCas} (Caseificio) $\subseteq$ $VR_{CodCas3}$ (Forma) VR causato dalla totalità dell'associazione diretta "Produce"

$VR_{CodCas3}$ (Forma) $\subseteq$ VR_{CodCas} (Caseificio) VR causato dalla totalità dell'associazione inversa "E'Prodotta"

f) mapping relazionale dell'associazione binaria "Acquista" tra le entità "Acquirente" e "Forma" di molteplicità 1:N

Acquirente (CodA, Tipologia, Cognome, Nome, DataN, PIva, Denominazione, Indirizzo, Città, Cap)

Forma : relazione già mappata in precedenza

VR_{CodA} (Acquirente) $\subseteq$ VR_{CoA1} (Forma) VR causato dalla totalità dell'associazione diretta "Acquista"

g) mapping relazionale dei vincoli di integrità

diagramma ER		modello logico relazionale
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 (Aquirente) : (Tipologia IN (“PR”, “GR”, “GD”))
Vincolo intrarelazionale o interno su singola n-pla sul dominio di un attributo

V2: (…..) ⟹ V2 (Forma) : (Scelta IN (1, 2))
Vincolo intrarelazionale o interno su singola n-pla sul dominio di un attributo

V3: (…..) ⟹ V3 (Forma) : (Stagionatura IN (12, 24, 30, 36))
Vincolo intrarelazionale o interno su singola n-pla sul dominio di un attributo

V4: (…..) ⟹ V3 (Foto) : (TipoFile IN (“JPG”, “GIF”, “BMP”))
Vincolo intrarelazionale o interno su singola n-pla sul dominio di un attributo

V5: (…..) ⟹ V3 (Forma) : (Venduta IN (“SI”, “NO”))
Vincolo intrarelazionale o interno su singola n-pla sul dominio di un attributo

Occorre effettuare infine (in questo caso non sarebbe necessario in quanto non presenti) il mapping relazionale degli eventuali vincoli referenziali dovuti al mapping relazionale di una generica associazione di molteplicità N:N che diventano vincoli interrelazionali o esterni referenziali

PREMESSA teorica

*La teoria della **normalizzazione** ha come scopo quello di fornire metodi per progettare basi di dati senza anomalie. La realtà che si intende analizzare tramite una base di dati può avere più schemi, equivalenti fra loro,che la descrivono. Alcuni di questi schemi possono presentare degli inconvenienti (ridondanze o anomalie di comportamento) che rendono inadeguato l’utilizzo efficiente delle informazioni.*

*La teoria della normalizzazione si occupa di definire criteri di bontà per schemi relazionali (**forme normali**)*

Una forma normale è una proprietà di uno schema relazionale che ne garantisce la “qualità”, cioè l’assenza di determinati difetti) ed eventualmente di determinare metodi algoritmici per ottenere uno schema “migliore” ed equivalente a partire da uno schema non in forma normale (normalizzazione)

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

4. 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, laddove presenti

N.B. Innanzitutto ipotizziamo di avere creato nel server MYSQL un utente che abbia tutti i privilegi e le autorizzazioni per operare sul database in oggetto.

```
CREATE DATABASE ConsorzioDB;
USE ConsorzioDB;
CREATE DOMAIN MiaTipologia AS CHAR (2) NOT NULL
    CHECK (VALUE IN (“PR”, “GR”, “GD”));                    //vincolo V1
CREATE DOMAIN MiaScelta AS INT (1) NOT NULL
    CHECK (VALUE IN (1, 2));                                //vincolo V2
CREATE DOMAIN MiaStagionatura AS INT (2) NOT NULL
    CHECK (VALUE IN (12, 24, 30, 36));                      //vincolo V3
CREATE DOMAIN MioTipoFile AS CHAR (3) NOT NULL
    CHECK (VALUE IN (“JPG”, “GIF”, “BMP”));                 //vincolo V4
CREATE DOMAIN MiaVenduta AS CHAR (2) NOT NULL
    CHECK (VALUE IN (“SI”, “NO”));                          //vincolo V5
CREATE TABLE Consorzio
  (
  CodCon                VARCHAR (10) NOT NULL,
  Denominazione         VARCHAR (50) NOT NULL
  Data Fondazione       DATE NOT NULL,
  Sede Legale           VARCHAR (255) NOT NULL,
  Sito Web              VARCHAR (255) NOT NULL,
  Email                 VARCHAR (30) NOT NULL,
  PRIMARY KEY (CodCon)
  ) Engine = InnoDB;
CREATE TABLE Utenza
  (
  CodCas                INT (4) NOT NULL,
  Password              VARCHAR (8) NOT NULL,
  ACLStringa            VARCHAR (20) NOT NULL
  PRIMARY KEY (CodCas)
  ) Engine = InnoDB;
CREATE TABLE Caseificio
  (
  CodCas                INT (4) NOT NULL,
  Denominazione         VARCHAR (50) NOT NULL,
  Titolare              VARCHAR (50) NOT NULL,
  Regione               VARCHAR (30) NOT NULL,
  Provincia             VARCHAR (2) NOT NULL,
  Indirizzo             VARCHAR (100) NOT NULL,
  Città                 VARCHAR (30) NOT NULL,
  Cap                   CHAR(5) NOT NULL,
  Email                 VARCHAR (30) NOT NULL,
  Telefono              VARCHAR (30) NOT NULL,
  Latitudine            VARCHAR (20) NOT NULL,
  Longitudine           VARCHAR (20) NOT NULL,
  CodCon1               VARCHAR (10) NOT NULL,
  PRIMARY KEY (CodCas),
  FOREIGN KEY (CodCon1) REFERENCES Consorzio (CodCon)
    ON DELETE CASCADE
    ON UPDATE CASCADE
  ) Engine = InnoDB;
```

```
CREATE TABLE RaccoltaLatte
  (
  CodR                    VARCHAR (10) NOT NULL,
  QtaLavorata             DECIMAL (8,2) NOT NULL,
  QtaFormaggio            DECIMAL (8,2) NOT NULL,
  DataRaccolta            DATE NOT NULL,
  CodCas1                 INT (4) NOT NULL,
  PRIMARY KEY (CodR),
  FOREIGN KEY (CodCas1) REFERENCES Caseificio (CodCas)
    ON DELETE CASCADE
    ON UPDATE CASCADE
  ) Engine = InnoDB;
CREATE TABLE Foto
  (
  CodImg                  VARCHAR (10) NOT NULL,
  Didascalia              VARCHAR (30) NOT NULL,
  PathFile                VARCHAR (200) NOT NULL,
  NomeFile                VARCHAR (30) NOT NULL,
  TipoFile                MioTipoFile,
  Progressivo             INT (2) NOT NULL,
  CodCas2                 INT (4) NOT NULL,
  PRIMARY KEY (CodImg),
  FOREIGN KEY (CodCas2) REFERENCES Caseificio (CodCas)
    ON DELETE CASCADE
    ON UPDATE CASCADE
  ) Engine = InnoDB;
CREATE TABLE Acquirente
  (
  CodA                    VARCHAR (10) NOT NULL,
  Tipologia               MiaTipologia,
  Cognome                 VARCHAR (30) NOT NULL,
  Nome                    VARCHAR (30) NOT NULL,
  DataN                   DATE NOT NULL
  PIva                    VARCHAR (20),
  Denominazione           VARCHAR (50),
  Indirizzo               VARCHAR (50),
  Citta                   VARCHAR (30),
  Cap                     CHAR (5),
  PRIMARY KEY (CodA),
  ) Engine = InnoDB;
CREATE TABLE Forma
  (
  CodF                    VARCHAR (10) NOT NULL,
  Venduta                 MiaVenduta,
  Stagionatura            MiaStagionatura,
  Scelta                  MiaScelta,
  Progressivo             INT (4) NOT NULL
  Peso                    DECIMAL (6,2) NOT NULL,
  DataProduzione          DATE NOT NULL,
  Prezzo                  DECIMAL (6,2),
  Data Acquisto           DATE,
  CodCas3                 INT (4) NOT NULL,
  CodA1                   VARCHAR (10) NOT NULL,
  PRIMARY KEY (CodF),
  FOREIGN KEY (CodCas3) REFERENCES Caseificio (CodCas)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (CodA1) REFERENCES Acquirente (CodA)
    ON DELETE CASCADE
    ON UPDATE CASCADE
  ) Engine = InnoDB;
```

5. le seguenti interrogazioni espresse in linguaggio SQL

PREMESSA_Nelle formulazione delle seguenti query è stata utilizzata la notazione sintattica [parametro] per indicare valori parametrici da inserire al momento della esecuzione delle medesime. Questo approccio (supportato ad esempio da MsAccess) dovrà essere adeguato se necessario alle caratteristiche del DBMS che sarà effettivamente utilizzato (ad esempio MYSQL non lo supporta).

a) Visualizzare il numero di forme prodotte da ciascun caseificio tra due date fornite

```
SELECT Caseificio.CodCas, COUNT(Forma.CodF) AS TotaleForme
FROM Caseificio, Forma
WHERE (Caseificio.CodCas = Forma.CodCas3) AND
            (Caseificio.DataProduzione BETWEEN [DataIni] AND [DataFine])
GROUP BY Caseificio.CodCas;
```

b) Visualizzare la media del latte lavorato giornalmente nell'anno corrente dai caseifici provincia per provincia

```
SELECT Caseificio.Provincia, Caseificio.CodCas, AVG (RaccoltaLatte.QtaLAvorata) AS MediaLatte
FROM Caseificio, RaccoltaLatte
WHERE (Caseificio.CodCas = Forma.CodCas1) AND
            (YEAR(Raccolta Latte.DataRaccolta) = YEAR(CURDATE( ))
GROUP BY Caseificio.CodCas, Caseificio.Provincia;
```

c) Visualizzare i dati del caseificio che ha venduto il maggior numero di forme di prima scelta in un anno impostato dall'utente

```
CRETE VIEW t1 (SELECT Caseificio.CodCas AS Codice, COUNT(Forma.CodF) AS TotaleForme
                FROM Caseificio, Forma
               WHERE (Caseificio.CodCas = Forma.CodCas3) AND (Forma.Venduta = "SI")
                        AND (Forma.Scelta = 1) AND (YEAR(Forma.DataAcquisto) = [AnnoX])
              ORDER BY COUNT(Forma.CodF) DESC);

SELECT *
FROM Caseificio
WHERE Caseificio.CodCas IN (SELECT t1.Codice
                            FROM t1
                           WHERE t1.TotaleForme = (SELECT MAX(t1.TotaleForme)
                                                   FROM t1) );
```

d) Visualizzare l'elenco dei caseifici che, in un certo periodo individuato da due date fornite dall'utente, hanno venduto meno di 10 forme di seconda scelta

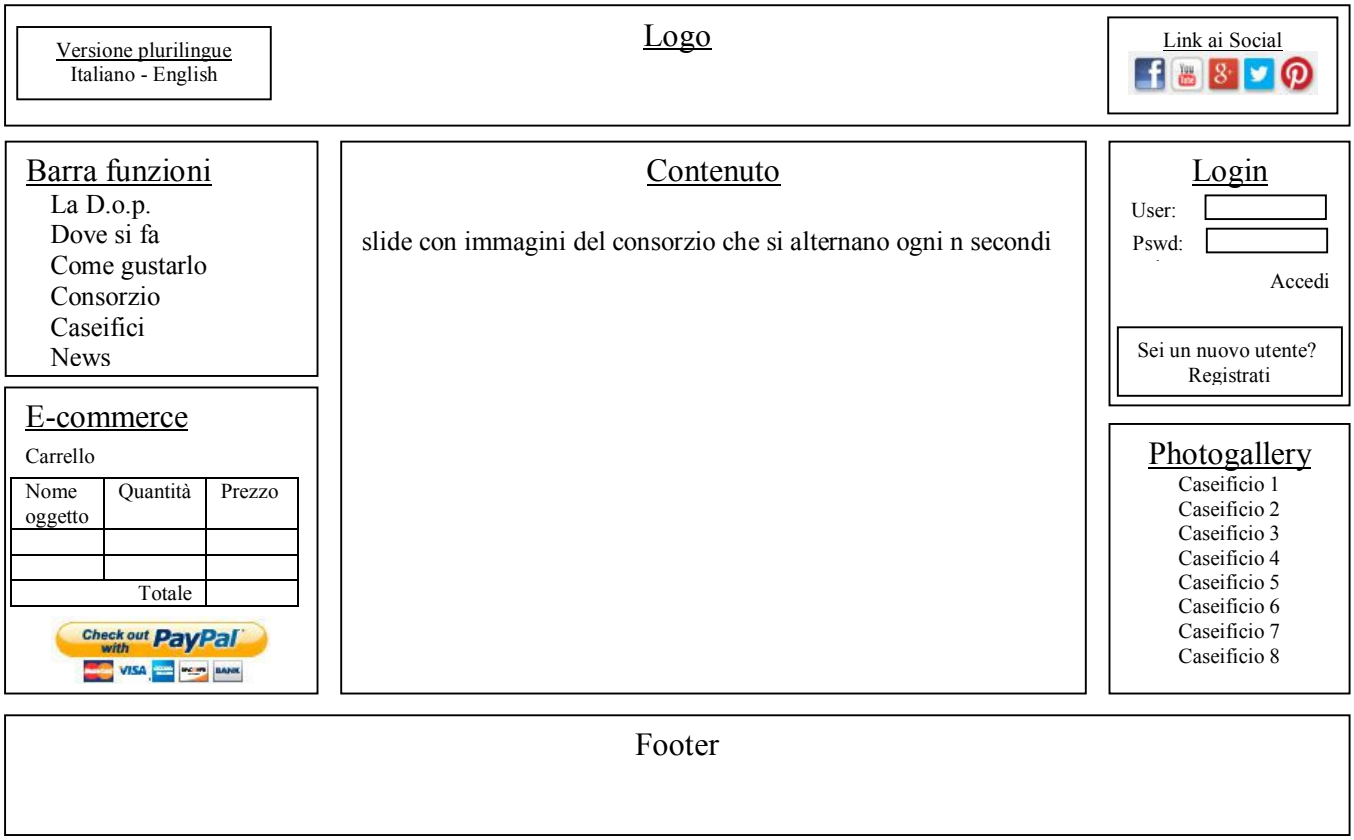
```
SELECT Caseificio.CodCas, Caseificio.Denominazione, COUNT(Forma.CodF) AS TotaleForme
FROM Caseificio, Forma
WHERE (Caseificio.CodCas = Forma.CodCas3) AND (Forma.Venduta = "SI") AND (Forma.Scelta = 2)
      AND (Forma.DataAcquisto BETWEEN [DataIni] AND [DataFine])
GROUP BY Caseificio.CodCas, Caseificio.Denominazione
HAVING COUNT(Forma.CodF) < 10;
```

6. Il progetto della Home page dell'interfaccia WEB che si intende proporre per la gestione del DataBase e delle gallerie di immagini dei luoghi di produzione

Date le caratteristiche e la natura del consorzio che probabilmente non dispone di personale specializzato per la gestione di un server per ospitare il sito web, la soluzione più indicata è quella di acquistare un servizio di *hosting* del sito da un fornitore di servizi web (spesso è possibile acquistare il servizio di *hosting* del sito insieme alla registrazione del dominio che in questo caso potrebbe essere www.consorzioformaggi.it): questa soluzione è senz'altro la migliore tenuto conto che il sito deve essere operativo 24 ore su 24 e 7 giorni su 7, ed è anche economica.

È necessario acquistare, oltre allo spazio web per ospitare il sito, l'integrazione dell'interprete PHP nel server web e la disponibilità di un database MySQL per la gestione del database: si tratta in ogni caso di un pacchetto molto comune nell'offerta dei fornitori di servizi web.

Da un punto di vista grafico la home page di tale sito potrebbe presentare il logo del consorzio, semplice e rappresentativo, disegnato con stile minimalista ma d'impatto.
La figura che segue rappresenta schematicamente la struttura della pagina web iniziale del consorzio.



Riguardo l'accessibilità al sito web si prevede una parte pubblica ed una privata:

- nella parte pubblica si hanno le risorse (foto, indirizzi dei caseifici, statistiche, news) ed alcune funzioni (mailTo, iscriviti alla newsletter);
- nella parte privata, a cui si accede una volta effettuato il login, a seconda dei permessi concessi ai vari utenti, alla barra delle funzioni vengono aggiunte dinamicamente altre funzionalità (aggiorna dati, amministra, elimina record, etc) cui ognuno di loro è abilitato.

Nel caso si voglia implementare una eventuale attività di e-commerce offrendo la possibilità di acquisto online dei prodotti, sarà necessario ridefinire alcuni elementi del database per organizzare una politica di gestione dei possibili clienti e dovrà contenere i link alla pagina aziendale del consorzio presente sui principali social network.

Questi, dopo essersi registrati, potrebbero accedere a specifiche funzionalità (acquista, carrello, vai al pagamento, ecc.) offerta da una apposita barra di menu.

Infine il footer della pagina potrebbe contenere i riferimenti legali del consorzio ed un piccolo riferimento al web-designer.

Per permettere l'accesso al sito anche tramite dispositivi mobili in modo che le pagine web si adattino automaticamente all'hardware utilizzato, la tecnologia utilizzata sarà di tipo *Responsive*. Col termine *Responsive Design* si indica un approccio per il quale la progettazione e lo sviluppo di un sito dovrebbero adattarsi al comportamento e all'ambiente dell'utente in base a fattori come le dimensioni dello schermo, la piattaforma e l'orientamento del device.

7. la codifica in un linguaggio a scelta di un segmento significativo dell'applicazione Web che consente l'interazione con la base di dati

Come esempio di pagina web dinamica che costituisce il sito del consorzio presentiamo lo script PHP **medialatte.php** che realizza il comportamento richiesto per la query (b).
Si è ipotizzato che il server MySQL sia in esecuzione sullo stesso computer del web server e che l'accesso al database "ConsorzioDB" venga effettuato come utente "root" dotato di una password precedentemente assegnata; inoltre, per semplicità, implementeremo la pagina web in oggetto completamente priva di struttura grafica.

```
<html>
  <head>
  <title>
      Media giornaliera latte per provincia anno corrente
  </title>
  </head>

  <body>
  <?php
  $connection = mysqli_connect("localhost", "root", <password assegnata>, "ConsorzioDB") or
       die("Errore di connessione al DBMS MySQL. . mysql_error());

  mysql_select_db("ConsorzioDB")
      or die("Selezione del database non riuscita" . mysql_error());

  $query = " SELECT Caseificio.Provincia, Caseificio.CodCas,
                    AVG (RaccoltaLatte.QtaLAvorata) AS MediaLatte
             FROM Caseificio, RaccoltaLatte
             WHERE (Caseificio.CodCas = Forma.CodCas1) AND
             (YEAR(Raccolta Latte.DataRaccolta) = YEAR(CURDATE( ))
             GROUP BY Caseificio.CodCas, Caseificio.Provincia;";

  $risultato = mysqli_query($connection, $query) or die("Query fallita" . mysql_error());

  $numrows = mysqli_num_rows($risultato);
  if ($numrows != 0)
     {
     echo "
     <table border>
     <caption><b>
        Media latte giornaliera per provincia anno corrente
     </b></caption>
       <tr>
         <td><b>Provincia</b></td>
         <td><b>Codice Caseificio</b></td>
         <td><b>Media Latte Lavorato (litri) </b></td>
       </tr>
         ";
     for ($i=0;$i<$numrighe;$i++)
        {
         $provincia = mysql_result($risultato, $i, 'Provincia');
         $codice = mysql_result($risultato, $i, 'CodCas');
         $media = mysql_result($risultato, $i, 'MediaLatte');
          echo "<tr>
                 <td><$provincia></td>
                 <td><$codice></td>
                 <td><$media></td>
               </tr>";
        }
     echo "</table>";
     }
  else
     {
     echo "Nessun dato trovato.";
     }

  mysql_free_result($risultato)
       or die("Liberazione risorse fallita" . mysql_error());

  mysql_close($connessione) or die("Chiusura connessione fallita" . mysql_error());

  ?>
</body>
</html>
```

Soluzione seconda parte

N.B. si ricorda che la traccia chiede di rispondere a 2 soli dei 4 quesiti proposti

I. In relazione al tema proposto nella prima parte, indichi come intende affrontare la gestione degli accessi riservati agli operatori dei caseifici per lo svolgimento delle loro funzioni.

Per quanto riguarda la gestione degli accessi riservati agli utenti dei caseifici, tenendo presente che questi opereranno attraverso un applicativo suddiviso in moduli ognuno dei quali espleta una specifica funzione, è possibile pensare ad una soluzione in cui:
- venga definita una tabella degli utenti in cui oltre allo username e password si memorizzi una stringa di caratteri ognuno dei quali fa riferimento ad uno specifico modulo e possa valere 1 o 0 a seconda che l'utente sia abilitato o meno all'uso della specifica funzionalità di quel modulo con una policy di tipo *ACL* (Access Control List)
- il software di interfaccia al momento della login di un utente memorizzi la stringa *ACL* in modo da consentire l'accesso ai soli moduli a cui l'utente è abilitato.

Questo modo di operare offre, sia pur gestita via software, una modalità operativa molto più flessibile di quanto non offra il meccanismo di *GRANT/REVOKE* offerto dai DBMS.
Per quanto riguarda le password degli operatori si potrebbe anche ipotizzare che queste non siano memorizzate in chiaro ma tramite una codifica hash (ad esempio MD5 supportata in PHP) in modo tale da renderne sicuro l'utilizzo in ambiente internet.

II. In relazione al tema proposto nella prima parte, sviluppi la query SQL per calcolare la percentuale di forme di seconda scelta prodotte annualmente da un certo caseificio (sul totale delle forme da lui prodotte annualmente).

```
SELECT Forme_secondascelta*100/Forme_totali As PercentualeSecondaScelta
FROM (SELECT COUNT(*) AS Forme_secondascelta
        FROM Caseificio, Forma
        WHERE (Caseificio.CodCas = Forma.CodCas3) AND (Forma.Scelta = 1)
           AND (Caseificio.CodCAs =[IdCaseificioX]) )AS t1,
       (SELECT COUNT(*) AS Forme_totali
       FROM Caseificio, Forma
           AND (Caseificio.CodCAs =[IdCaseificioX]) )AS t2,
```

III. Illustri, anche servendosi di esempi, il concetto di "vista" in una base di dati.

N.B. Basta ricordare quanto fatto a lezione

Una **vista** è una **relazione** che non è fisicamente memorizzata sulla base di dati (dove invece sono fisicamente memorizzate tutte le relazioni create attraverso l'istruzione di **CREATE TABLE**) ottenuta tramite un'opportuna operazione di **"mapping"** (mappatura) con le tabelle effettivamente memorizzate.
Dopo essere stata creata, una vista può essere **modificata** ed **interrogata**.

Per creare una vista si utilizza l'istruzione SQL **CREATE VIEW** la cui sintassi è la seguente:

CREATE VIEW <NomeTabellaVista> **AS** <Query> **;**

Dove:
- <NomeTabellaVista> è il nome assegnato alla fittizia tabella della vista;
- <Query> è una normale query formulata con l'istruzione **SELECT**.

Come si intuisce il motivo per cui si creano le **viste** è quello di fornire ad un gruppo di utenti **una versione semplificata o parziale** di una realtà che può essere molto più complessa.
Categorie diverse di utenti possono interagire con la base di dati utilizzando il loro punto di vista e trascurando quelli degli altri.

E' importante ricordare che ogni modifica apportata su di una tabella vista si ripercuote sulla/e tabella/e dalla quale è stata tratta.

Per eliminare una vista si utilizza l'istruzione SQL **DROP VIEW** la cui sintassi è:

DROP VIEW <NomeTabellaVista> **;**

Sulle viste è possibile utilizzare l'istruzione **GRANT**.

IV. Descriva le tipologie di linguaggi utilizzate in un DBMS.

N.B. Basta ricordare quanto fatto a lezione

L'utente usa un **DBMS** attraverso appropriati **comandi** grazie ai quali viene instaurata una **comunicazione** con il sistema di elaborazione che gestisce la basi di dati o database.
L'insieme di tali comando costituisce un vero e proprio linguaggio, anzi costituisce **un insieme di vari linguaggi**

- **DDL** (**D**ata **D**efinition **L**anguage ossia **Linguaggio di definizione dei dati**).
E' il linguaggio utilizzato per la descrizione dei dati, delle tabelle, delle interfacce utente e delle *viste* (sottoschemi). Permette all'utente di costruire la struttura fisica del database partendo dallo schema logico.

- **DMCL** (**D**evice **M**edia **C**ontrol **L**anguage ossia **Linguaggio di controllo dei supporti di memorizzazione dei dati**).
E' il linguaggio che permette alla struttura fisica del database di far riferimento alle particolari unità di memoria di massa utilizzate dal sistema.

- **DML** (**D**ata **M**anipulation **L**anguage ossia **Linguaggio per il trattamento dei dati**).
E' il linguaggio utilizzato per accedere al database per effettuare inserimenti, modiche e cancellazioni.

- **DCL** (**D**ata **C**ontrol **L**anguage ossia **Linguaggio di controllo dei dati**)
E' il linguaggio utilizzato per stabilire i vincoli di integrità, oltre che per stabilire accessi e permessi.

- **QL** (**Q**uery **L**anguage ossia **Linguaggio di interrogazione dei dati**)
E' il linguaggio interattivo che permette di interrogare il database per ritrovare le informazioni relative alla chiave di ricerca impostata dall'utente.

La diffusione attuale del modello relazionale ha favorito la nascita del linguaggio per basi di dati ossia di un insieme di comandi che consentono la gestione globale del database raggruppando in se le funzioni dei linguaggi DDL, DMCL, DML, DCL e QL.

Il più importante di tutti è **l'SQL** (***S****tructured* ***Q****uery* ***L****anguage*) che è un linguaggio **non procedurale** (in quanto non richiede la descrizione dei passi elementari di elaborazione) **o di tipo dichiarativo** (in quanto è un tipo di linguaggio in cui le istruzioni si limitano a descrivere "cosa" si vuol fare e non "come" farlo).

E' ormai da tempo uno degli standard tra i linguaggi per basi di dati relazionali.

Il linguaggio **SQL** può essere usato in:

- **modalità stand-alone**: in questa modalità può essere classificato come **query language interattivo.** I comandi vengono inviati al sistema operativo in modo **interattivo** (utilizzando un'apposita interfaccia grafica con menù, finestre ed icone) **oppure batch** (creando file di istruzioni da eseguire in gruppo).
In entrambe le modalità viene invocato **l'interprete** SQL.
Le interrogazioni o query vengono composte utilizzando le operazioni dell'algebra relazionale ottenendo come risultato sempre una tabella che rappresenta una relazione.

- **modalità embedded**: in questa modalità è possibile utilizzare comandi SQL all'interno di istruzioni di altri linguaggi (esempio C, Java, C++) detti "linguaggi ospite".
In linea di massima un programma scritto in *linguaggio ospite compilato* (che incorpora comandi SQL al suo interno) subirà un primo processo di *precompilazione* (che si occupa di tradurre le sole istruzioni SQL in istruzioni in linguaggio ospite compilato) seguito da una vera e propria *compilazione* (che si occupa di tradurre le istruzioni specifiche del linguaggio ospite).