Relazione sul progetto di Basi di Dati

Project Museum

Francesco Antonio Migliorin, Matricola 1096825

Davide Saonara, Matricola 1094288

Indice

1. Abstract

Il comune Padova è una delle città più ricche di storia in Italia e meta ambita dal turismo culturale mondiale per il suo patrimonio storico, artistico e religioso. Per avere una visione più chiara dei suoi musei storici e artistici, sotto la decisione del consiglio comunale, la giunta ci ha incaricato un database per gestirli.
L'applicazione deve fornire ai cittadini i dati generali dei musei e di tutti i reperti presenti nella città sia quelli esposti che quelli in magazzino, indicandone il periodo nel quale è previsto il restauro.
Un Direttore di museo deve poter accedere alla lista delle generalità dei dipendenti, compresi i loro dati sensibili, e anche alla lista dei fornitori, potendo risalire a chi ha venduto o donato un dato reperto a uno dei musei.

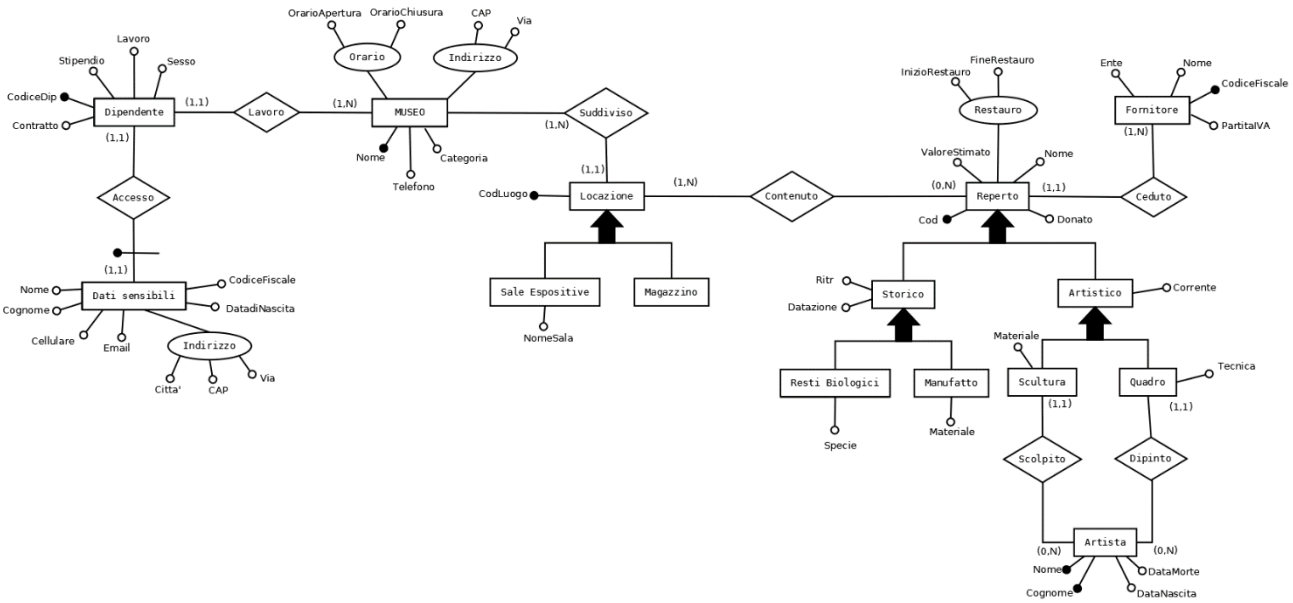
2. Descrizione Dei Requisiti

Si vuole realizzare una base di dati per la gestione dei musei storici e artistici della provincia di Padova.

L'intera provincia conta trentasette **Musei,** devono essere registrate le informazioni riguardanti nome, la categoria a cui appartengono, l'indirizzo, il Cap, telefono e l'orario di apertura e chiusura, contenenti reperti di gran valore storico o artistico, e dove lavorano dei **Dipendenti**, identificati da un codice univoco, di cui va riportato lo stipendio, la tipologia di lavoro e di contratto, e il sesso. I restanti dati quali nome, cognome, codice fiscale, data di nascita, cellulare, email e indirizzo vengono considerati **Dati Sensibili**. Ogni museo ha al suo interno un'area utilizzata per depositare, al sicuro, i reperti non esposti o quelli che stanno venendo restaurati: il **Magazzino**. I restanti reperti invece vengono esposti in delle **Sale Espositive**, che normalmente hanno un nome. Entrambi questi luoghi sono identificati da un codice. L'entità principale che si va quindi a modellare è **Reperti**; di loro è importante conoscere il nome, la data di inizio e di fine del restauro, il valore stimato, se è stato donato e il codice che lo identifica univocamente. Un modo interessante di distinguere i reperti è trattarli come **Storici** o **Artistici**. Nei primi è fondamentale conoscere la datazione e il luogo di ritrovamento e si dividono ulteriormente in **Resti Biologici,** dove si deve conoscere la specie di appartenenza, memorizzata in nomenclatura binomiale, e **Manufatti** dov'è rilevante sapere il materia di cui è composto. Per quanto riguarda i secondi vengono suddivisi in **Sculture**, dove si deve riportare il materiale con il quale sono fatte, e in **Quadri**, in cui si illustra la tecnica pittorica utilizzata. Ogni reperto artistico è stato dipinto o scolpito da un **Artista** che ha un nome, un cognome e, se note, una data di nascita e di morte. C'è inoltre un **Fornitore** associato a ogni reperto, identificato dal suo codice fiscale, di cui è necessario fornire nome, ente e partita iva, nel caso non fosse un privato.

3. Progettazione Concettuale

3.1 Schema Concettuale E-R



3.2 Lista delle classi

Tutti gli attributi delle classi sono NOT NULL poiché sono elementi indispensabili per la gestione della base di dati, con nessuna ridondanza. I soli attributi che possono assumere valore NULL sono: ValoreStimato, InzioRestauro, FineRestauro, DataNascita, DataMorte, NomeSala, Cellulare, Email
Tutti gli attributi classificati come int vengono considerati come degli interi senza segno, altrimenti la parte negativa rimarrebbe inutilizzata da parte della base di dati.

- **Reperto**: modella un generico reperto
  Attributi:
  - Cod: Int <<PK>> - codice identificativo e chiave primaria
  - Nome: String – Nome del reperto
  - Donato: Enum ('SI', 'NO') – coppia di caratteri che identifica se il reperto è stato donato (SI) o venduto (NO)
  - ValoreStimato: Int – Stima del valore del reperto espressa in €
  - InizioRestauro: date – Indica la data di inizio dei lavori di restauro
  - InizioFine: date – Indica la data di fine dei lavori di restauro

  Sono definite le seguenti sottoclassi di Reperto (con vincolo partizionamento):
  1. **Storico:** modella un reperto storico
     Attributi:
     - Datazione: Int – Età approssimativa del reperto
     - Ritr: String – Paese dove il reperto è stato portato alla luce

     Sono definite le seguenti sottoclassi di Storico (con vincolo partizionamento):
     1) **Resto Biologico:** modella un reperto biologico di una qualche specie
        Attributi:
        - Specie: String – Specie di appartenenza del resto biologico, scritta in nomenclatura binomiale
     2) **Manufatto**: modella un insieme di reperti creati da una civiltà

Attributi:
- Materiale: String – Specifica i materiale del quale è composto il manufatto

2. **Artistico**: modella un reperto artistico
   Attributi:
   - Corrente: String – Specifica il nome della corrente artistica del reperto artistico

   Sono definite le seguenti sottoclassi di Artistico (con vincolo partizionamento):
   1) **Scultura**: modella un reperto scolpito in un materiale
      Attributi:
      - Materiale: String – Specifica il materiale di cui la scultura è composta
   2) **Quadro**: modella un reperto dipinto con vari colori
      Attributi:
      - Tecnica: String – Specifica il nome della tecnica utilizzata per la realizzazione

- **Artista**: modella un generico artista
  Attributi:
  - Nome: String – Nome dell'artista
  - Cognome: String – Cognome dell'artista
  - DataNascita: date – Specifica la data di nascita se risaputa
  - DataMorte: date – Specifica la data di morte se risaputa
  Nome e Cognome insieme formano la chiave primaria, inoltre un artista per essere tale deve avere creato almeno una scultura o un quadro

- **Fornitore**: modella un generico fornitore
  Attributi:
  - CodiceFiscale: String <<PK>> - codice fiscale e chiave primaria
  - Nome: String – Nome del fornitore
  - Ente: Enum ('SI', 'NO') – coppia di caratteri che identifica se il fornitore è un ente (SI) o un privato (NO)
  - PartitaIVA: String – Codice della partita IVA dell'azienda
  Ogni Reperto ha e deve avere un solo Fornitore, inoltre solo i fornitori che sono enti devono avere la partita iva

- **Locazione**: modella un generico luogo
  Attributi:
  - CodLuogo: Int <<PK>> - codice identificativo e chiave primaria
  Sono definite le seguenti sottoclassi di Locazione (con vincolo partizionamento):
  1. **Sala Espositiva**: modella una sala di esposizione reperti
     Attributi:
     - NomeSala: String – Il nome di una sala
     Una Sala Espositiva deve avere almeno un reperto
  2. **Magazzino**: modella un deposito di reperti
     Ogni Museo ha un solo Magazzino

- **Museo**: modella un generico museo
  Attributi:
  - Nome: String <<PK>> - Il nome proprio del museo
  - Categoria: String – Stringa contenente la tipologia del museo
  - Via: String – Stringa contenente il nome della via e il numero civico del museo
  - CAP: String – Cap della città di residenza del museo
  - Telefono: String – Numero telefonico del museo
  - OrarioApertura: time – Indica la data di apertura del museo
  - OrarioChiusura: time – Indica la data di chiusura del museo
  Un Museo per essere definito tale deve avere almeno una Sala Espositiva

- **Dipendente**: modella un generico dipendente
  Attributi:
  - CodiceDip: Int <<PK>> - codice identificativo e chiave primaria che si auto incrementa

- Stipendio: Int – Compenso mensile del lavoratore
- Contratto: Enum ('Part Time', 'Full Time') – Identifica la tipologia di contratto del dipendente, e le relative ore di lavoro settimanali: non più di 30 per un Part Time e 38 per un Full Time.
- Lavoro: String – Impiego all'interno del museo
- Sesso: Enum ('M', 'F') – carattere che identifica se il lavoratore è donna (F) o uomo (M)

- **Dati Sensibili**: modella una generica collezione di dati sensibili
  Attributi:
  - CodiceFiscale: String - codice fiscale
  - Nome: String – Nome del soggetto
  - Cognome: String – Cognome del soggetto
  - Cellulare: String – Numero telefonico del soggetto
  - Email: String – indirizzo di posta elettronica del soggetto
  - Via: String – Stringa contenente il nome della via e il numero civico del soggetto
  - Città: String – Nome della città di residenza del soggetto
  - CAP: String – Cap della città di residenza del soggetto
  - DatadiNascita: date – Indica la data di nascita del soggetto

3.3 Lista delle associazioni

- Dati sensibili - Dipendente: Accesso
  - Ogni dipendente ha accesso a dei dati sensibili
- Dipendente - Museo: Lavoro
  - Ogni dipendente ha un lavoro in un dato Museo
  - Il Museo ha almeno un dipendente che ci lavora
- Museo - Locazione: Suddiviso
  - Ogni museo è suddiviso in diverse Locazioni
  - La singola Locazione è una parte di un singolo Museo
- Locazione - Reperto: Contenuto
  - Ogni Locazione contiene differenti reperti
  - Il singolo Reperto può essere contenuto in una locazione
- Reperto - Fornitore: Ceduto
  - Ogni Reperto viene ceduto da un fornitore
  - Il Fornitore cede uno o più reperti
- Scultura - Artista: Scolpito
  - Ogni Scultura viene scolpita da un artista
  - L'artista può scolpire uno o più sculture
- Quadro - Artista: Dipinto
  - Ogni Quadro viene dipinto da un artista
  - L'Artista può dipingere uno o più quadri

4. Progettazione Logica

4.1 Gerarchie

La gerarchia di Locazione viene semplificata grazie all'accorpamento delle entità figlie in quella padre. Questa operazione è possibile, e conveniente, perché gli accessi al padre e alle figlie avvengono simultaneamente, o quasi. Vengono quindi eliminate le classi Sale Espositive e Magazzino. Il risultato della ristrutturazione è il seguente:

Locazione: eredita l'attributo NomeSala proveniente da Sale Espositive.

È importante notare come la classe Locazione presenti ora anche un attributo Tipo per poter differenziare comunque la tipologia.

La gerarchia di Storico viene semplificata grazie all'accorpamento delle entità figlie in quella padre, come spiegato sopra. Vengono dunque eliminate le classi Resti Biologici e Manufatto. Rimane, quindi, solo una classe:

Storico: eredita gli attributi di Resti Biologici e di Manufatto, rispettivamente Specie e Materiale

Viene implementato anche in questo caso un attributo Tipo per poter di distinguere la differenza

La gerarchia Artistico viene semplificata grazie all'accorpamento delle entità figlie in quella padre. A differenza delle altre illustrate in questo caso troviamo due associazioni che, in seguito all'eliminazione delle classi figlie, vengono correlate alla classe padre: Artistico.
Il risultato è come segue:

Artistico: essa eredita non solo gli attributi di Scultura e di Quadro, cioè Materiale e Tecnica, ma anche le associazioni Scolpito e Dipinto, che ora vengono rappresentate da un'unica relazione Creato.

In questo caso, come sopra, viene aggiunta un attributo Tipo.

La gerarchia di Reperto viene tradotta tramite l'accorpamento del genitore nelle entità figlie.
L'operazione, anche definita come collasso verso il basso, è realizzabile perché la gerarchia è totale ed esclusiva, quindi non vi sono problemi di ridondanza. Viene, dunque, eliminata la relazione Reperto e vengono rinominate le classi restanti:

Reperto Artistico: Prende tutti gli attributi di Reperto, cioè Cod, Nome, Descrizione, Donato, Valore Stimato, InizioRestauro e FineRestauro

Reperto Storico: Prende tutti gli attributi di Reperto, cioè Cod, Nome, Descrizione, Donato, Valore Stimato, InizioRestauro e FineRestauro

4.2 Chiavi Sintetiche

Per agevolare una certa flessibilità nella stesura del codice e semplificare la gestione della base di da ti sono state adottate le seguenti chiavi sintetiche: CodiceMuseo all'entità museo, ID all'entità fornitore e IDArtista all'entità artista. Quest'ultima è stata inserita nell'eventualità che si presentino due artisti aventi lo stesso nome e cognome.

4.3 Schema Relazionale

LEGENDA: CHIAVE PRIMARIA, CHIAVE ESTERNA

Museo(CodiceMuseo, Nome, Categoria, Via, CAP, Telefono, OrarioApertura, OrarioChiusura)

Dipendente(CodiceDip, Stipendio, Contratto, Lavoro, Sesso, CodiceFiscale, Nome, Cognome, Cellulare, Email, Via, Città, Cap, DatadiNascita, CodMuseo)

Locazione(CodLuogo, NomeSala, Tipologia ,CodMuseo)

RepertoStorico(CodS, Nome, Donato, ValoreStimato, InizioRestauro, FineRestauro, Datazione, Ritr, SpecApp, Materiale, Tipo, CodFornitore, CodDep)

RepertoArtistico(CodA, Nome, Donato, ValoreStimato, InizioRestauro, FineRestauro, Corrente, Materiale, Tecnica, Tipo, CodFornitore, CodDep)

Creato(CodReperto, CodArtista)

Artista(IDArtista, Nome, Cognome, DataNascita, DataMorte)

Fornitore(ID, Nome, Ente, CodiceFiscale, PartitaIVA)

VINCOLI DI INTEGRITÀ

Vincolo di integrità referenziale tra l'attributo CodMuseo di Dipendente e l'attributo CodiceMuseo di Museo

Vincolo di integrità referenziale tra l'attributo CodMuseo di Locazione e l'attributo CodiceMuseo di Museo

Vincolo di integrità referenziale tra l'attributo CodDep di RepertoStorico e l'attributo CodLuogo di Locazione

Vincolo di integrità referenziale tra l'attributo CodFornitore di RepertoStorico e l'attributo ID di Fornitore

Vincolo di integrità referenziale tra l'attributo CodDep di RepertoArtistico e l'attributo CodLuogo di Locazione

Vincolo di integrità referenziale tra l'attributo CodFornitore di RepertoArtistico e l'attributo ID di Fornitore

Vincolo di integrità referenziale tra l'attributo CodReperto di Creato e l'attributo CodA di RepertoArtistico

Vincolo di integrità referenziale tra l'attributo CodArtista di Creato e l'attributo IDArtista di Artista

5. Implementazione dello schema logico

I vincoli di integrità del tipo “Per A deve esistere almeno un B” non sono stati implementati visto la loro scarsa importanza e le parecchie righe di codice necessarie per la loro implementazione per la creazione del record principale e del primo record correlato. Mentre l’implementazione è stata realizzata in modo tale che sia possibile creare separatamente un record principale e un numero arbitrario di record secondari. Il codice qui di seguito definisce le tabelle: MUSEO, DIPENDENTE, FORNITORE, LOCAZIONE, ARTISTA, REPERTOARTISTICO, REPERTOSTORICO, CREATO, ERRORI. La tabella ERRORI è stata implementata per registrare quando viene generato un errore relativo ai trigger. Visto che la base di dati imposterà il valore di errore in NULL, segnalerà anche dove è stata applicata la correzione oltre che alla descrizione dell’errore.

```
1.  DROP TABLE IF EXISTS Creato;
2.  DROP TABLE IF EXISTS RepertoArtistico;
3.  DROP TABLE IF EXISTS RepertoStorico;
4.  DROP TABLE IF EXISTS Artista;
5.  DROP TABLE IF EXISTS Locazione;
6.  DROP TABLE IF EXISTS Fornitore;
7.  DROP TABLE IF EXISTS Dipendente;
8.  DROP TABLE IF EXISTS Museo;
9.  DROP TABLE IF EXISTS Errori;
10.
11. CREATE TABLE IF NOT EXISTS Errori
12. (
13. CodiceErrore smallint UNSIGNED NOT NULL AUTO_INCREMENT,
14. Descrizione varchar(255) NOT NULL,
15. PRIMARY KEY (CodiceErrore)
16. )ENGINE=InnoDB;
17.
18. /*Crea la tabella Museo*/
19. CREATE TABLE IF NOT EXISTS Museo
20. (
21. CodiceMuseo tinyint UNSIGNED NOT NULL AUTO_INCREMENT,
22. Nome varchar(200) NOT NULL,
23. Categoria varchar(100) NOT NULL,
24. Via varchar(250) NOT NULL,
25. CAP varchar(5) NOT NULL,
26. Telefono varchar(15) NOT NULL,
27. OrarioApertura time NOT NULL,
28. OrarioChiusura time NOT NULL,
29. PRIMARY KEY (CodiceMuseo)
30. ) ENGINE=InnoDB;
31.
32. /*Crea la tabella Dipendente*/
33. CREATE TABLE IF NOT EXISTS Dipendente
34. (
35. CodiceDip smallint UNSIGNED NOT NULL AUTO_INCREMENT,
36. Stipendio int UNSIGNED NOT NULL DEFAULT 0,
37. Contratto ENUM('Part Time', 'Full Time'),
38. Lavoro varchar(100) NOT NULL,
39. Sesso ENUM('M', 'F'),
40. CodiceFiscale varchar(16) NOT NULL,
41. Nome varchar(25) NOT NULL,
42. Cognome varchar(25) NOT NULL,
43. Cellulare varchar(20) DEFAULT NULL,
44. Email varchar(50) DEFAULT NULL,
45. Via varchar(250) NOT NULL,
46. Citta varchar(100) NOT NULL,
47. CAP varchar(5) NOT NULL,
48. DatadiNascita date NOT NULL,
```

```
CodMuseo tinyint UNSIGNED NOT NULL,
PRIMARY KEY(CodiceDip),
FOREIGN KEY (CodMuseo) REFERENCES Museo(CodiceMuseo) ON DELETE CASCADE
) ENGINE=InnoDB;

/*Crea la tabella Fornitore*/
CREATE TABLE IF NOT EXISTS Fornitore
(
ID smallint UNSIGNED NOT NULL AUTO_INCREMENT,
Nome varchar(100) NOT NULL,
Ente ENUM ('SI','NO') NOT NULL,
CodiceFiscale varchar(16) NOT NULL,
PartitaIVA varchar(11) DEFAULT NULL,
PRIMARY KEY (ID)
) ENGINE=InnoDB;

/*Crea la tabella Locazione*/
CREATE TABLE IF NOT EXISTS Locazione
(
CodLuogo smallint UNSIGNED NOT NULL AUTO_INCREMENT,
NomeSala varchar(50) NOT NULL,
Tipologia varchar(50) NOT NULL,
CodMuseo tinyint UNSIGNED NOT NULL,
PRIMARY KEY (CodLuogo),
FOREIGN KEY (CodMuseo) REFERENCES Museo(CodiceMuseo) ON DELETE CASCADE
)ENGINE=InnoDB;

/*Crea la tabella Artista*/
CREATE TABLE IF NOT EXISTS Artista
(
IDArtista smallint UNSIGNED NOT NULL AUTO_INCREMENT,
Nome varchar(25) NOT NULL,
Cognome varchar(25) NOT NULL,
DataNascita date null DEFAULT null,
DataMorte date null DEFAULT null,
PRIMARY KEY (IDArtista)
)ENGINE=InnoDB;

/*Crea la tabella RepertoArtistico*/
CREATE TABLE IF NOT EXISTS RepertoArtistico
(
CodA smallint UNSIGNED NOT NULL,
Nome varchar(50) NOT NULL,
Donato ENUM ('SI','NO') NOT NULL,
ValoreStimato bigint UNSIGNED,
InizioRestauro date DEFAULT NULL,
FineRestauro date DEFAULT NULL,
Corrente varchar(25) NOT NULL,
Materiale char(50) DEFAULT NULL,
Tecnica char(50) DEFAULT NULL,
Tipo char(20) NOT NULL,
        CodFornitore smallint UNSIGNED DEFAULT NULL,
        CodDep smallint UNSIGNED DEFAULT NULL,
        PRIMARY KEY (CodA),
        FOREIGN KEY (CodFornitore) REFERENCES Fornitore(ID) ON DELETE SET NULL,
        FOREIGN KEY (CodDep) REFERENCES Locazione(CodLuogo) ON DELETE SET NULL
        ) ENGINE=InnoDB;

        /*Crea la tabella RepertoStorico*/
        CREATE TABLE IF NOT EXISTS RepertoStorico
        (
        CodS smallint UNSIGNED NOT NULL,
        Nome varchar(50) NOT NULL,
        Donato ENUM ('SI','NO') NOT NULL,
        ValoreStimato bigint UNSIGNED,
        InizioRestauro date DEFAULT NULL,
        FineRestauro date DEFAULT NULL,
```

```
116.        Datazione bigint DEFAULT NULL,
117.        Ritr varchar(50) NOT NULL,
118.        SpecApp varchar(50) DEFAULT NULL,
119.        Materiale varchar(50) DEFAULT NULL,
120.        Tipo varchar(20) NOT NULL,
121.        CodFornitore smallint UNSIGNED DEFAULT NULL,
122.        CodDep smallint UNSIGNED DEFAULT NULL,
123.        PRIMARY KEY (CodS),
124.        FOREIGN KEY (CodFornitore) REFERENCES Fornitore(ID) ON DELETE SET NULL,
125.        FOREIGN KEY (CodDep) REFERENCES Locazione(CodLuogo) ON DELETE SET NULL
126.        ) ENGINE=InnoDB;
127.
128.        /*Crea la tabella Creato*/
129.        CREATE TABLE IF NOT EXISTS Creato
130.        (
131.        CodReperto smallint UNSIGNED NOT NULL,
132.        CodArtista smallint UNSIGNED NOT NULL,
133.        PRIMARY KEY (CodReperto,CodArtista),
134.        FOREIGN KEY (CodReperto) REFERENCES RepertoArtistico(CodA) ON DELETE CASCADE,
135.        FOREIGN KEY (CodArtista) REFERENCES Artista(IDArtista) ON DELETE CASCADE
136.        )ENGINE=InnoDB;
```

6. Trigger, Funzioni, Query e Procedure

Qui di seguito verranno riportati i codici relativi ad alcuni Trigger, Funzioni, Query e Procedure ed alcuni esempi relative a quest'ultime due.

6.1 Trigger

Sono stati implementati alcuni trigger passivi per il corretto mantenimento corretto della base di dati:

- **ControlloDipIns** e **ControlloDipUp**: Verificano che il codice fiscale di un dipendente sia scritto in maiuscolo altrimenti provvede a correggerlo. Il primo si attiva in caso di INSERT mentre il secondo in caso di UPDATE.
  Codice:

```
1.  DELIMITER $
2.  DROP TRIGGER IF EXISTS ControlloDipIns$
3.
4.  CREATE TRIGGER ControlloDipIns
5.  BEFORE INSERT ON Dipendente
6.  FOR EACH ROW
7.  BEGIN
8.  SET NEW.CodiceFiscale=UPPER(NEW.CodiceFiscale);
9.  END$
10.
11. DROP TRIGGER IF EXISTS ControlloDipUp$
12.
13. CREATE TRIGGER ControlloDipUp
14. BEFORE UPDATE ON Dipendente
15. FOR EACH ROW
16. BEGIN
17. SET NEW.CodiceFiscale=UPPER(NEW.CodiceFiscale);
18. END$
```

- **ControlloForIns** e **ControlloForUp**: Verificano che il codice fiscale e la partita IVA, nel caso non sia nulla, di un fornitore siano scritte in maiuscolo altrimenti provvede a correggerle. Inoltre nel caso il fornitore sia un privato controlla che non si tenti di inserire una partita IVA.

Nel caso che ciò dovesse verificarsi imposta il valore della partita IVA a NULL e inserisce nell'apposita tabella l'errore e dove si è verificato. Il primo si attiva in caso di INSERT mentre il secondo in caso di UPDATE.
Codice:

```
        DROP TRIGGER IF EXISTS ControlloForIns$

CREATE TRIGGER ControlloForIns
BEFORE INSERT ON Fornitore
FOR EACH ROW
BEGIN
SET NEW.CodiceFiscale=UPPER(NEW.CodiceFiscale);
IF NEW.PartitaIVA IS NOT NULL THEN
    IF NEW.Ente=2 THEN
        SET NEW.PartitaIVA=NULL;
        INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il Fornitore Codice Fiscale: ', NEW.CodiceFiscale,' ha generato un errore nella tabella Fornitori. Un privato non ha partita IVA ! '));
    ELSE SET NEW.PartitaIVA=UPPER(NEW.PartitaIVA);
    END IF;
END IF;
END$

DROP TRIGGER IF EXISTS ControlloForUp$

CREATE TRIGGER ControlloForUp
BEFORE UPDATE ON Fornitore
FOR EACH ROW
BEGIN
SET NEW.CodiceFiscale=UPPER(NEW.CodiceFiscale);
IF NEW.PartitaIVA IS NOT NULL THEN
    IF NEW.Ente=2 THEN
        SET NEW.PartitaIVA=NULL;
        INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il Fornitore Codice Fiscale: ', NEW.CodiceFiscale,' ha generato un errore nella tabella Fornitori. Un privato non ha partita IVA ! '));
    ELSE SET NEW.PartitaIVA=UPPER(NEW.PartitaIVA);
    END IF;
END IF;
END$
```

- **ControlloDataReArtIns** e **ControlloDataReArtUP**: Verifica che la data di fine restauro non si antecedete a quella di inizio restauro nei Reperti Artistici. Nel caso che ciò dovesse verificarsi imposta le due date a NULL e inserisce nell'apposita tabella l'errore e dove si è verificato. Il primo si attiva in caso di INSERT mentre il secondo in caso di UPDATE.
  Codice:

```
        DROP TRIGGER IF EXISTS ControlloDataReArtIns$

CREATE TRIGGER ControlloDataReArtIns
BEFORE INSERT ON RepertoArtistico
FOR EACH ROW
BEGIN
IF CURDATE() > NEW.InizioRestauro THEN
SET NEW.InizioRestauro=NULL;
    SET NEW.FineRestauro=NULL;
    INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodA,' ha generato un errore nella tabella RepertoArtistico. InizioRestauro antecedente a oggi'));
ELSE IF NEW.FineRestauro < NEW.InizioRestauro THEN
    SET NEW.InizioRestauro=NULL;
    SET NEW.FineRestauro=NULL;
```

```
14.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodA,' ha gener
    ato un errore nella tabella RepertoArtistico. InizioRestauro > FineRestauro ! '));
15.     END IF;
16. END IF;
17. END$
18.
19. DROP TRIGGER IF EXISTS ControlloDataReArtUP$
20.
21. CREATE TRIGGER ControlloDataReArtUP
22. BEFORE UPDATE ON RepertoArtistico
23. FOR EACH ROW
24. BEGIN
25. IF CURDATE() > NEW.InizioRestauro THEN
26. SET NEW.InizioRestauro=NULL;
27.     SET NEW.FineRestauro=NULL;
28.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodA,' ha gener
    ato un errore nella tabella RepertoArtistico. InizioRestauro antecedente a oggi'));
29. ELSE IF NEW.FineRestauro < NEW.InizioRestauro THEN
30.     SET NEW.InizioRestauro=NULL;
31.     SET NEW.FineRestauro=NULL;
32.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodA,' ha gener
    ato un errore nella tabella RepertoArtistico. InizioRestauro > FineRestauro ! '));
33.     END IF;
34. END IF;
35. END$
```

- **ControlloDataReStoIns** e **ControlloDataStoUP**: analogo al precedente, solo che si riferisce ai Reperti Storici invece che agli Artistici.
  Codice:

```
1.         DROP TRIGGER IF EXISTS ControlloDataReStoIns$
2.
3.  CREATE TRIGGER ControlloDataReStoIns
4.  BEFORE INSERT ON RepertoStorico
5.  FOR EACH ROW
6.  BEGIN
7.  IF CURDATE() > NEW.InizioRestauro THEN
8.  SET NEW.InizioRestauro=NULL;
9.      SET NEW.FineRestauro=NULL;
10.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodS,' ha gener
    ato un errore nella tabella RepertoStorico. InizioRestauro antecedente a oggi'));
11. ELSE IF NEW.FineRestauro < NEW.InizioRestauro THEN
12.     SET NEW.InizioRestauro=NULL;
13.     SET NEW.FineRestauro=NULL;
14.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodS,' ha gener
    ato un errore nella tabella RepertoStorico. InizioRestauro > FineRestauro ! '));
15.     END IF;
16. END IF;
17. END$
18.
19. DROP TRIGGER IF EXISTS ControlloDataStoUP$
20.
21. CREATE TRIGGER ControlloDataStoUP
22. BEFORE UPDATE ON RepertoStorico
23. FOR EACH ROW
24. BEGIN
25. IF CURDATE() > NEW.InizioRestauro THEN
26. SET NEW.InizioRestauro=NULL;
27.     SET NEW.FineRestauro=NULL;
28.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodS,' ha gener
    ato un errore nella tabella RepertoStorico. InizioRestauro antecedente a oggi'));
29. ELSE IF NEW.FineRestauro < NEW.InizioRestauro THEN
30.     SET NEW.InizioRestauro=NULL;
31.     SET NEW.FineRestauro=NULL;
```

```
32.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Il reperto ID: ', NEW.CodS,' ha gener
    ato un errore nella tabella RepertoStorico. InizioRestauro > FineRestauro ! '));
33.     END IF;
34. END IF;
35. END$
```

- **ControlloDataArtistaIns** e **ControlloDataArtistaUP:** Verifica che la data di morte di un artista non sia antecedente a quella di nascita. Nel caso che ciò dovesse verificarsi imposta le due date a NULL e inserisce nell'apposita tabella l'errore e dove si è verificato. Il primo si attiva in caso di INSERT mentre il secondo in caso di UPDATE.
  Codice:

```
1.  DROP TRIGGER IF EXISTS ControlloDataArtistaIns$
2.
3.  CREATE TRIGGER ControlloDataArtistaIns
4.  BEFORE INSERT ON Artista
5.  FOR EACH ROW
6.  BEGIN
7.  IF NEW.DataMorte < NEW.DataNascita THEN
8.      SET NEW.DataMorte=NULL;
9.      SET NEW.DataNascita=NULL;
10.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Errore generato da ID: ', NEW.Nome, '
     ', New.Cognome, ' nella tabella Artista. DataMorte > DataNascita ! '));
11. END IF;
12. END$
13.
14. DROP TRIGGER IF EXISTS ControlloDataArtistaUP$
15.
16. CREATE TRIGGER ControlloDataArtistaUP
17. BEFORE UPDATE ON Artista
18. FOR EACH ROW
19. BEGIN
20. IF NEW.DataMorte < NEW.DataNascita THEN
21.     SET NEW.DataMorte=NULL;
22.     SET NEW.DataNascita=NULL;
23.     INSERT INTO Errori(Descrizione) VALUES (CONCAT ('Errore generato da ID: ', NEW.Nome, '
     ', New.Cognome, ' nella tabella Artista. DataMorte > DataNascita ! '));
24. END IF;
25. END$
26.
27. DELIMITER ;
```

6.2 Funzioni

6.2.1 Funzione che ritorna un booleano se nel museo scelto è presente il reperto scelto

```
1.  DROP FUNCTION IF EXISTS RepertoEsposto;
2.  DELIMITER $
3.  CREATE FUNCTION RepertoEsposto (nome_reperto VARCHAR(50), nome_museo VARCHAR(200))
4.  RETURNS bit
5.  BEGIN
6.              DECLARE temp SMALLINT UNSIGNED;
7.              SELECT Count(CodS) INTO temp
```

```
            FROM Museo M, Locazione L, RepertoArtistico RA
WHERE M.Nome=nome_museo AND RA.Nome=nome_reperto AND M.CodiceMuseo=L.CodMuseo AND L.CodLuo
go=RA.CodDep;
            IF temp>0 THEN
                RETURN 1;
            ELSE
                RETURN 0;
            END IF;
END$
```

6.2.2 Funzione che ritorna il numero di giorni alla fine del restauro dato il codice del Reperto Artistico

```
DELIMITER $
CREATE FUNCTION GiorniRestaurazioneArtistico ( codice SMALLINT UNSIGNED )
RETURNS int
BEGIN
    DECLARE temp SMALLINT UNSIGNED;
    SELECT DATEDIFF(CURDATE(),FineRestauro) INTO temp
    FROM RepertoArtistico
    WHERE CodA=codice;
    RETURN TEMP;
END$
DELIMITER ;
```

6.3 Query

6.3.1 Mostrare il nome e il cognome di tutti i dipendenti con il guadagno più alto rispetto a quelli che lavorano a Padova

```
SELECT  Dip.Nome, Dip.Cognome, Citta
FROM Dipendente Dip
WHERE Dip.Stipendio>(SELECT MAX(Stipendio) FROM Dipendente WHERE Citta='Padova');
```

Il risultato della query sulla nostra base di dati è:

Nome	Cognome	Citta
Carla	Iadanza	Casalserugo
Giuseppe	Mazzi	Casalserugo
Fulgenzia	Monaldo	Casalserugo
Lorenzo	Schiavone	Noventa Padovana
Italia	Greco	Noventa Padovana
Pasquale	Cocci	Albignasego

6.3.2 Trovare tutti i reperti storici donati che valgono più di tutti i reperti comprati, Visualizzare Codice, Nome e Valore Stimato

```
SELECT CodS AS Codice, Nome, ValoreStimato AS Valore
FROM RepertoStorico
```

```
3. WHERE Donato=2 AND ValoreStimato>(SELECT MAX(ValoreStimato) FROM RepertoStorico WHERE Dona
   to=1);
```

Il risultato della query sulla nostra base di dati è:

Codice	Nome	Valore
13	scheletro di un umano	3512
111	Statuina	3000

6.3.3 Trovare tutti i dipendenti maschi nati negli anni ’50 che hanno lo stesso CAP del museo

```
1. SELECT Dipendente.Nome
2. FROM Dipendente,Museo
3. WHERE Dipendente.CAP=Museo.CAP AND YEAR(Dipendente.DatadiNascita) between 1950 and 1960
   AND Sesso=1 ;
```

Il risultato della query sulla nostra base di dati è:

Nome
Ermenegildo
Alessio
Salvatore
Uranio
Ireneo
Adone
Nazario
Guerrino
Ulisse

6.3.4 Trovare tutte le lavoratrici femmine che guadagnano più del valore medio dei manufatti presenti a Padova.

```
1. DROP VIEW IF EXISTS MediaValoreS;
2. SET SQL_MODE='';
3. CREATE VIEW MediaValoreS AS
4.     SELECT AVG(ValoreStimato) AS Media
5.     FROM RepertoStorico
6.     WHERE Tipo='Manufatto';
7. SELECT Dip.Cognome, Dip.Nome, Dip.Stipendio
8. FROM Dipendente Dip, Museo M, Locazione L, RepertoStorico R
9. WHERE Dip.CodMuseo=M.CodiceMuseo AND M.CodiceMuseo=L.CodMuseo AND L.CodLuogo=R.CodDep AND
   Dip.Sesso='F'
10. GROUP BY Dip.Cognome, Dip.Nome
11. HAVING Dip.Stipendio>(Select * FROM MediaValoreS);
```

Il risultato della query sulla nostra base di dati è:

Cognome	Nome	Stipendio
Boni	Isotta	2574
Buccho	Abelina	3091
Buccho	Anna	2081
Calabrese	Ninfa	5090
Calabresi	Norma	2850
Cocci	Quintina	3173
Conti	Cirilla	2237
Cremonesi	Natalina	2596
Esposito	Silvia	2237
Fallaci	Marzia	4000
Fanucci	Marianna	2647
Ferrari	Noemi	2093
Genovese	Rosaura	2062
Greco	Italia	5527
Iadanza	Annibale	3703
Iadanza	Carla	5364
Lorenzo	Urania	2574
Lucciano	Nilde	3315
Mancini	Vincenza	3226
Marchesi	Alvisa	2586
Mazzanti	Penelope	2051
Monaldo	Fulgenzia	5906
Napolitani	Lidia	3471
Napolitano	Agnese	2086
Nucci	Candida	2228
Nucci	Pasqualina	4603
Panicucci	Wanda	3487
Pisano	Giada	3659
Ricci	Flavia	4371
Ricci	Luigina	3116
Rizzo	Giuliana	3920
Trentini	Edmonda	3367
Trevisani	Clorinda	3439
Trevisani	Cristina	2732
Zito	Blanda	2538
Zito	Bruna	3529

6.3.5 Mostrare il numero totale di reperti storici e artistici nei musei con il CAP 35121

```sql
DROP TABLE IF EXISTS table1;
DROP TABLE IF EXISTS table2;

CREATE TEMPORARY TABLE IF NOT EXISTS table1 AS (SELECT CodiceMuseo, M.Nome, COUNT(CodS) AS TotaleRepertiStorici
FROM Museo M, Locazione L, RepertoStorico RS
WHERE M.CodiceMuseo=L.CodMuseo AND L.CodLuogo=RS.CodDep AND M.CAP=35121
GROUP BY CodiceMuseo);

CREATE TEMPORARY TABLE IF NOT EXISTS table2 AS (SELECT CodiceMuseo, M.Nome, COUNT(CodA) AS TotaleRepertiArtistici
FROM Museo M, Locazione L, RepertoArtistico RA
WHERE M.CodiceMuseo=L.CodMuseo AND L.CodLuogo=RA.CodDep
GROUP BY CodiceMuseo);

SELECT table1.Nome AS 'Nome Museo', TotaleRepertiStorici AS 'Totale Reperti Storici', TotaleRepertiArtistici AS 'Totale Reperti Artistici'
FROM table1,table2
WHERE table1.CodiceMuseo=table2.CodiceMuseo;
```

Il risultato della query sulla nostra base di dati è:

Nome Museo	Totale Reperti Storici	Totale Reperti Artistici
MUSEO DELLA TERZA ARMATA	15	10
MUSME - MUSEO DI STORIA DELLA MEDICINA IN PADOVA	10	10
PALAZZO SANTO STEFANO	8	8
MUSEO DI ZOOLOGIA	10	10
MUSEO DI GEOLOGIA E PALEONTOLOGIA	4	2
MUSEI CIVICI AGLI EREMITANI	10	5
CAPPELLA DEGLI SCROVEGNI	6	3
CHIESA DEGLI EREMITANI	8	4
SCUOLA DI S. MARIA DELLA CARITÃ€	10	5

6.3.6 Mostrare il numero totale di reperti storici e artistici nei musei con il CAP 35121

```sql
SELECT MAX(RepertiEsposti) AS 'Reperti Esposti', MAX(RepertiInMagazzino) AS 'Reperti in Magazzino'
FROM (SELECT COUNT(CodS) as RepertiEsposti, 0 AS RepertiInMagazzino
        FROM Museo M, Locazione L, RepertoStorico R
        WHERE M.CodiceMuseo=L.CodMuseo AND L.CodLuogo=R.CodDep AND L.Tipologia<>'Magazzino' AND M.CAP=35121
        UNION ALL
        SELECT NULL AS RepertiEsposti, COUNT(CodS) as RepertiEsposti
        FROM Museo M, Locazione L, RepertoStorico R
        WHERE M.CodiceMuseo=L.CodMuseo AND L.CodLuogo=R.CodDep AND L.Tipologia='Magazzino' AND M.CAP=35121) T;
```

Il risultato della query sulla nostra base di dati è:

Reperti Esposti	Reperti in Magazzino
62	19

6.4 Procedure

6.4.1 Procedura che rimuove Reperto Artistico

```sql
DROP PROCEDURE IF EXIST RimuoviReperto;
DELIMITER $
CREATE PROCEDURE RimuoviReperto (IN CodiceReperto smallint UNSIGNED)
            BEGIN
            DECLARE Temp smallint UNSIGNED;
                SELECT CodA INTO Temp
                FROM RepertoArtistico
                WHERE CodA=CodiceReperto;
                IF Temp IS NOT NULL THEN
                    DELETE FROM RepertoArtistico WHERE CodA=Temp;
            END IF;
        END$
DELIMITER ;
```

6.4.2 Procedura che aumenta lo stipendio dato il Codice Identificativo del Direttore, del Dipendente e la percentuale dell'aumento

```sql
DROP PROCEDURE IF EXISTS AumentoStipendio;
DELIMITER $
CREATE PROCEDURE AumentoStipendio (IN CodiceDirettore smallint UNSIGNED, CodiceDipendente smallint UNSIGNED, percentuale int UNSIGNED)
    BEGIN
        DECLARE Temp smallint;
        SELECT CodiceDip INTO Temp
        FROM Dipendente
        WHERE CodMuseo=(SELECT CodMuseo FROM Dipendente Dir WHERE Dir.CodiceDip=CodiceDirettore AND Dir.Lavoro='direttore') AND CodiceDip=CodiceDipendente;
        IF(Temp=CodiceDipendente) THEN
            UPDATE Dipendente
            SET Stipendio=Stipendio+((Stipendio*percentuale)/100)
            WHERE CodiceDip=Temp AND Lavoro <> 'direttore';
        END IF;
    END$

DELIMITER ;
```