

PROGETTO BASI DI DATI I

CLASS DIAGRAM UML



STUDENTI:

DI LUCA GIUSEPPE N86002488

DI MARTINO CARMINE N86002553

DOCENTI:

PERON ADRIANO

DE LUCA ALESSANDRO

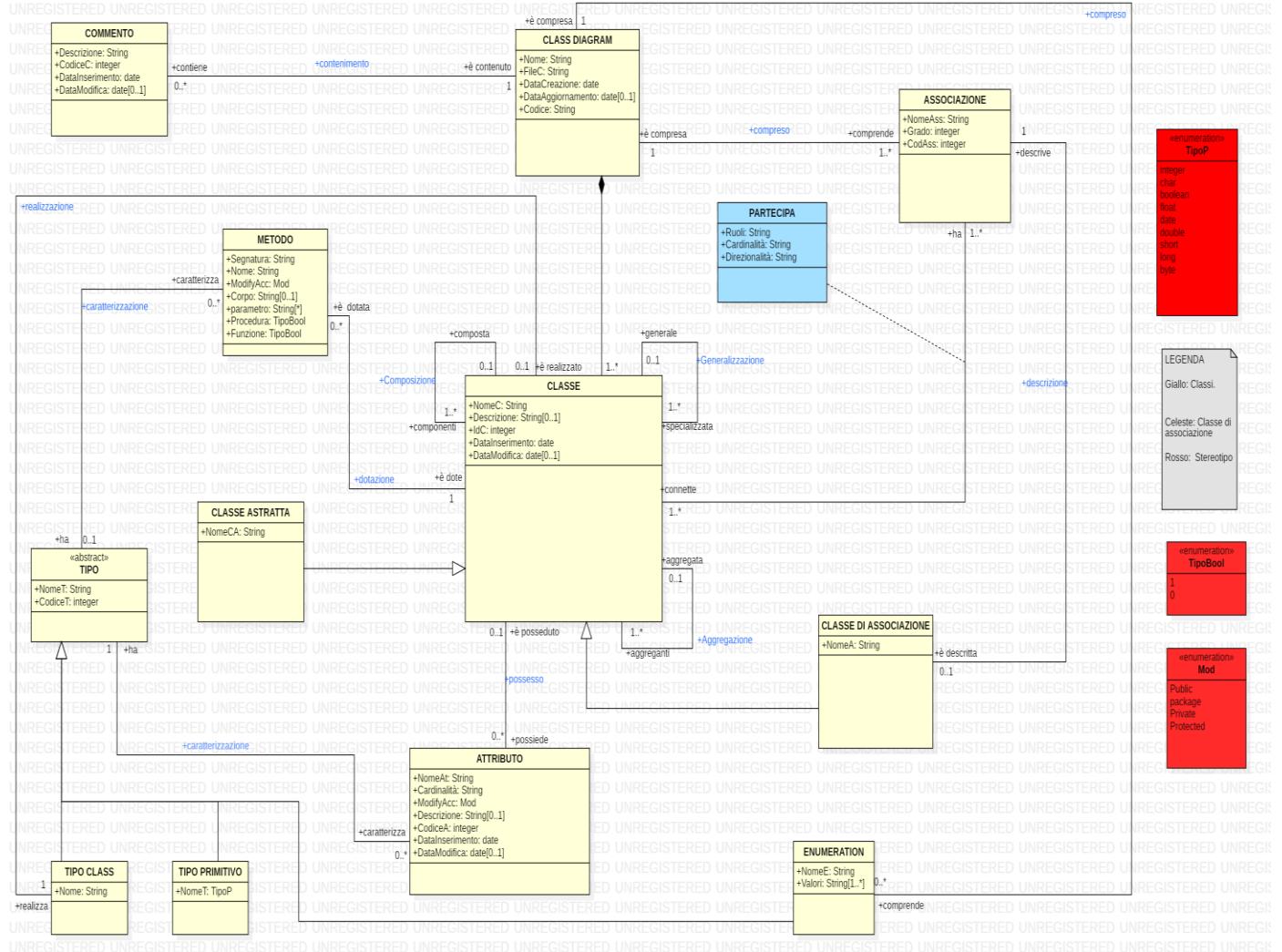
DESCRIZIONE DEL PROGETTO

Il progetto consiste nell'implementazione di una base di dati relazionale per la descrizione e memorizzazione di class diagram UML.

Nella prima fase del progetto vi è un modello teorico per la rappresentazione concettuale e grafica dei dati a un alto livello di astrazione, necessaria a tradurre le informazioni risultanti dall'analisi di un determinato dominio in uno schema concettuale rappresentato tramite un diagramma UML ai fini della rappresentazione dei dati e le relazioni che intercorrono tra esse. Dopodichè nella seconda fase, per descrivere in maniera corretta ed efficiente tutte le informazioni contenute nell' UML viene presentato uno schema logico, ma prima di passare allo schema logico, il diagramma UML va ristrutturato per soddisfare due esigenze: quella di semplificare la traduzione e quella di ottimizzare il progetto. In questa fase si delineeranno anche eventuali vincoli da imporre. Nella terza ed ultima fase, ovvero quella della progettazione fisica viene presentato la traduzione dello schema logico dei dati in uno **schema fisico dei dati** contenente le definizioni delle tabelle, dei relativi vincoli di integrità e l'implementazione delle **transazioni** in SQL.

Terminata questa fase la base di dati è stata completamente progettata e si è passato alla sua **realizzazione**, cioè alla costruzione fisica delle tabelle e all'implementazione delle applicazioni della base di dati. Le applicazioni sono scritte in linguaggi di programmazione ad alto livello (Java) che riutilizzano il codice SQL scritto per le transazioni.

CLASS DIAGRAM

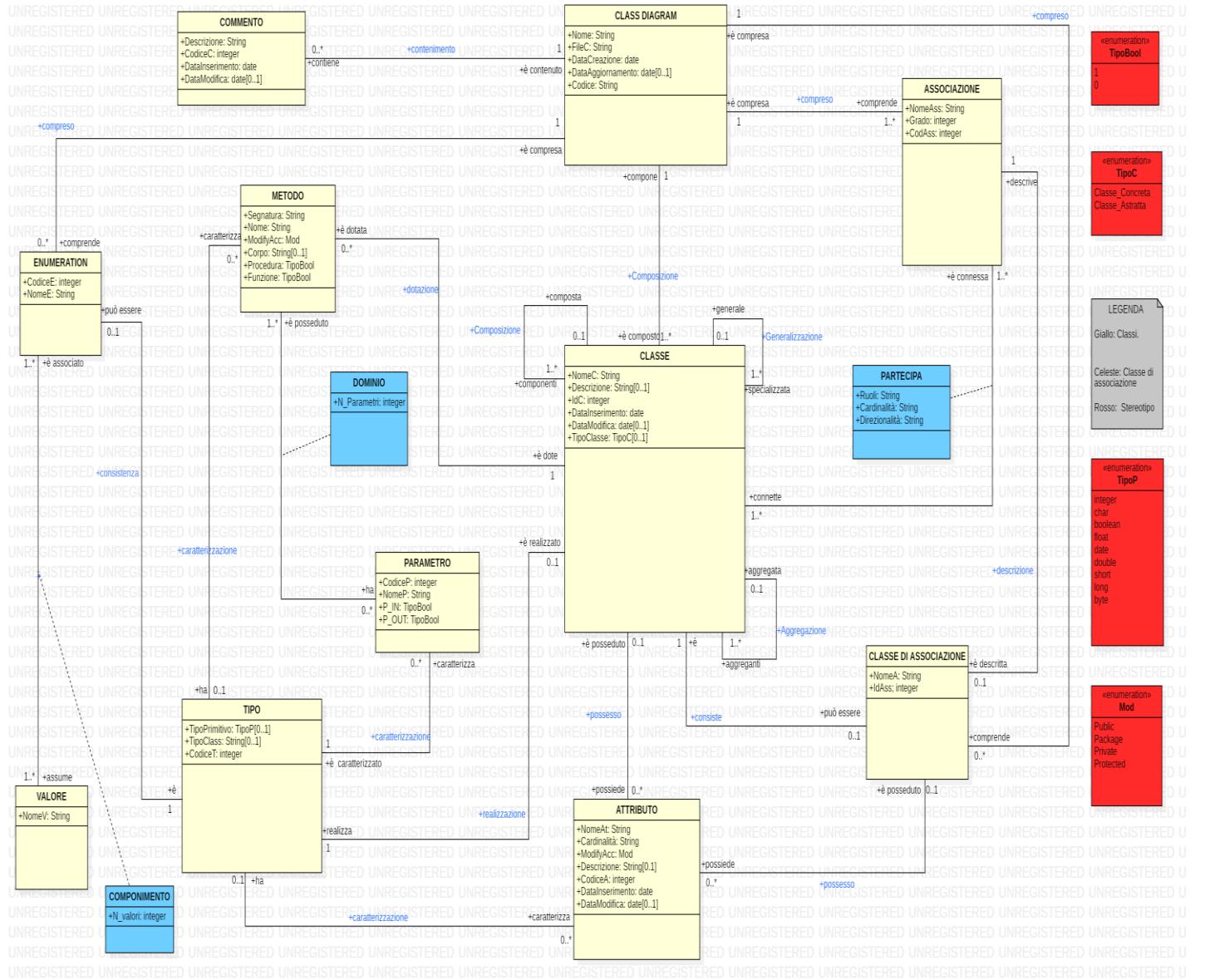


RISTRUTTURAZIONE DEL CLASS DIAGRAM

Per quanto riguarda la ristrutturazione del class diagram sono state apportate le seguenti modifiche:

- 1) Eliminazione della relazione composizione (tra “class” e “class diagram”) trasformata in associazione con opportuna molteplicità.
- 2) Accorpamento delle specializzate (“TipoPrimitivo” e “TipoClass”) nella generale (Tipo) con aggiunta del vincolo → VincoloTipo. Mentre per quanto riguarda la terza classe specializzata ossia “Enumeration”, si è ritenuto opportuno adottare il metodo conservativo ovvero trasformare la specializzata “Enumeration” in una classe normale, connettendola alla classe “tipo” scegliendo l’opportuna molteplicità e rispettando il vincolo.
- 3) Accorpamento della specializzata “classe astratta” nella generale “classe” introducendo un attributo discriminante (TipoClasse) che permette di selezionare il tipo della classe, ovvero “classe_concreta” o “classe astratta” (OR ESCLUSIVO), per quanto riguarda l’attributo (NomeCA) di “classe astratta” si è ritenuto opportuno eliminarlo per eliminare ridondanze in quanto nelle classe “classe” era già presente. Mentre invece per la classe “classe di associazione”, si è ritenuto opportuno trasformare la relazione di specializzazione (tra “classe” e “classe di associazione”) usando il metodo conservativo ossia trasformazione della specializzata in classe normale connettendola alla classe “classe” scegliendo l’opportuna molteplicità e rispettando il vincolo aggiunto → Scelta_Classe.
- 4) Eliminazione dell’attributo multiplo “valore” appartenente alla classe “Enumeration” e creazione della classe “Valore” connettendola tramite una relazione di associazione alla classe “Enumeration” usando l’opportuna molteplicità.
- 5) Eliminazione dell’attributo “parametro” appartenente alla classe “Metodo” e creazione della classe “Parametro” connettendola tramite una relazione di associazione alla classe “Metodo” usando l’opportuna molteplicità.

CLASS DIAGRAM RISTRUTTURATO



SCHEMA RELAZIONALE

CLASSE (IdC , NomeC, Descrizione , TipoClasse, DataInserimento, DataModifica , Codice , Generale , Aggregata , Composta, IdAss);

CHIAVI ESTERNE : Codice → CLASSDIAGRAM (Codice);
Generale → CLASSE (IdC);
Aggregata → CLASSE (IdC);
Composta → CLASSE (IdC);
IdAss → CLASSE_DI_ASSOCIAZIONE (IdAss);

ATTRIBUTO (CodiceA ,NomeAt , Cardinalità , DataInserimento, DataModifica, ModifyAcc , Descrizione , CodiceT , IdC , IdAss);

CHIAVI ESTERNE : CodiceT → TIPO (CodiceT);
IdC → CLASSE (IdC);
IdAss → CLASSE_DI_ASSOCIAZIONE (IdAss);

METODO (Segnatura , Nome , Corpo, Procedura , funzione, ModifyAcc , CodiceT , IdC);

CHIAVI ESTERNE : CodiceT → TIPO (CodiceT);
IdC → CLASSE (IdC);

TIPO (CodiceT , TipoPrimitivo , TipoClass , IdC , CodiceE);

CHIAVI ESTERNE : CodiceE → ENUMERATION (CodiceE);
IdC → CLASSE (IdC);

PARTECIPA (Ruoli , Cardinalità, Direzionalità , CodAss , IdC);

CHIAVI ESTERNE : CodAss → ASSOCIAZIONE (CodAss);
IdC → CLASSE (IdC);

CLASSE_DI_ASSOCIAZIONE (IdAss ,NomeA, Codice);

CHIAVI ESTERNE : Codice → CLASSDIAGRAM (Codice);

CLASS DIAGRAM (Codice , FileC, Nome, DataCrezione, DataAggiornamento);

COMMENTO (CodiceC , Descrizione , DataInserimento, DataModifica , Codice);

CHIAVI ESTERNE : Codice → CLASSDIAGRAM (Codice);

ASSOCIAZIONE (CodAss , NomeAss , Grado , Codice , IdAss);

CHIAVI ESTERNE : Codice → CLASSDIAGRAM (Codice);
IdAss → CLASSE_DI_ASSOCIAZIONE (IdAss);

ENUMERATION (CodiceE, NomeE , Codice);

CHIAVI ESTERNE : Codice → CLASSDIAGRAM (Codice);

VALORE (NomeV);

PARAMETRO (CodiceP , NomeP, P_IN , P_OUT, CodiceT);

CHIAVI ESTERNE : CodiceT → Tipo (CodiceT);

COMPONENTO (N_valori, NomeV, CodiceE);

CHIAVI ESTERNE : NomeV → VALORE (NomeV);
CodiceE → ENUMERATION (CodiceE);

DOMINIO (N_parametri , Segnatura, CodiceP);

CHIAVI ESTERNE : Segnatura → Metodo (Segnatura);
CodiceP → Parametro(CodiceP);

DIZIONARIO DELLE CLASSI

CLASSE	DESCRIZIONE	ATTRIBUTI	NOTE
CLASS DIAGRAM	Diagrammi che consentono di descrivere tipi di entità, con le loro caratteristiche e le eventuali relazioni tra questi tipi	Codice: String //Identificativo del class diagram Nome: String // nome del class diagram FileC: String // file contenente il class diagram DataCreazione: Date //Data della creazione del class diagram DataAggiornamento: Date[0..1] //Data di modifica del class diagram	Ogni class diagram ha un codice che lo identifica univocamente.
CLASSE	Una classe rappresenta una categoria di entità, ed è corredata da un insieme di attributi e operazioni	IdC: integer // id della classe NomeC: String // nome della classe Descrizione: String[0..1] // descrizione della classe DataInserimento: Date //Data di inserimento della classe DataModifica: Date [0..1] //Data di modifica della classe TipoClasse : TipoC //Indica il tipo della classe	Il nome della classe indica la categoria di entità descritta dalla classe. (ogni nome e Id della classe sono univoci). TipoClasse può assumere uno ed un solo valore tra: classe_concreta e classe_astratta.
ATTRIBUTO	Informazione che definisce lo stato interno dell'entità	CodiceA: integer // codice dell'attributo NomeAtt: String //nome dell'attributo (univoco in ogni classe) Cardinalità : String //indica la quantità di valori che può avere l'attributo per ogni istanza associata ModifyAcc: Mod // indica la visibilità	ModifyAcc può assumere uno ed un solo valore tra: package, protected, private, public. CodiceA è univoco per ogni attributo.

DIZIONARIO DELLE CLASSI

		<p>dell'attributo Descrizione: String[0..1] <i>//descrizione</i> dell'attributo DataInserimento: Date <i>//Data di inserimento</i> dell'attributo DataModifica: Date [0..1] <i>//Data di modifica</i> dell'attributo</p>	
TIPO	Definisce un insieme di valori che una variabile o un metodo possono assumere	<p>CodiceT: integer <i>//codice identificativo del tipo</i> TipoPrimitivo: TipoP [0..1] <i>//nome del tipo di tipo primitivo</i> TipoClass: String [0..1] <i>//nome del tipo di tipo class</i></p>	<p>La classe tipo può essere soltanto una tra TipoPrimitivo o TipoClass (OR esclusivo).</p> <p>Ogni tipo ha un codice che lo identifica univocamente.</p>

METODO	Operazioni o servizi offerti dall'oggetto all'esterno, realizzano il comportamento dell'oggetto	<p>Segnatura: String <i>//insieme di informazioni che identificano univocamente il metodo stesso</i> Nome: String <i>//Nome del metodo</i> ModifyAcc: Mod <i>//Indica la visibilità del metodo</i> Corpo: String <i>//Blocco di istruzioni contenuto nel metodo</i> Procedura: String <i>//specifica se il metodo è una procedura</i> Funzione: String <i>specifica se il metodo è una funzione</i></p>	<p>ModifyAcc può avere uno ed un solo valore tra: package, protected, private, public.</p>
COMMENTO	Nozione che specifica una caratteristica	<p>Descrizione: String <i>//Descrizione del commento</i> CodiceC: integer <i>//Codice identificativo del commento</i></p>	<p>Ogni commento ha un codice che lo identifica univocamente.</p>

DIZIONARIO DELLE CLASSI

		DataInserimento: Date //Data di inserimento del commento DataModifica: Date [0..1] //Data di modifica del commento	
CLASSE DI ASSOCIAZIONE	Classe derivata da un'associazione, utilizzata per aggiungere attributi ad un'associazione	IdAss: integer // identificativo della classe di associazione NomeA: String //Nome della classe di associazione (univoco)	Sottintendono un vincolo aggiuntivo, ossia che ci può essere solo un'istanza della classe di associazione fra ogni coppia di oggetti associati.
ASSOCIAZIONE	Indica che le istanze delle classi su cui sussiste l'associazione, condividono una relazione statica	NomeAss: String //Nome dell'associazione CodAss: integer //Codice identificativo dell'associazione Grado: integer //Specifica quante classi mette in relazione un'associazione	Ogni associazione ha un codice che la identifica univocamente.
PARTECIPA	Indica i ruoli e la cardinalità con cui le associazioni partecipano nella relazione tra le classi	Ruoli: String //indica il ruolo con la quale la classe partecipa all'associazione Cardinalità: String //numero di volte che una data istanza di entità deve o può partecipare alla associazione. Direzionalità: String //Indica chi ha la responsabilità di tenere traccia dell'associazione	
ENUMARATION	Rappresenta un insieme vincolato di tipi che può assumere un attributo	CodiceE: integer //Identificativo dell'enumeration NomeE: String //Indica il nome dell'enumeration	Ogni enumeration ha un codice che lo identifica univocamente.
VALORE	Rappresenta i valori che può assumere un enumeration	NomeV: String //Indica il nome del valore	Ogni valore è identificato univocamente da NomeV e CodiceE.

DIZIONARIO DELLE CLASSI

PARAMETRO	Specifica i parametri di input e di output di un metodo	<pre>CodiceP: integer //Identificativo del parametro NomeP: String //Indica il nome del parametro P_IN: String //Indica se il parametro è di tipo input P_OUT:String //Indica se il parametro è di tipo output</pre>	Ogni Parametro è identificato univocamente da un codice.
COMPONENTO	Definisce un insieme di valori che una enumerazione può assumere	<pre>N_Valori: integer //Indica il numero di valori che una enumerazione contiene</pre>	
DOMINIO	Definisce un insieme di parametri che un metodo possiede	<pre>N_parametri: integer //Indica il numero di parametri che un metodo possiede</pre>	

DIZIONARIO DELLE ASSOCIAZIONI

ASSOCIAZIONE	DESCRIZIONE	CLASSI	ATTRIBUTI
CONTENIMENTO	Un class diagram potrebbe avere commenti al suo interno.	COMMENTO [1] Ruolo: è contenuto CLASS DIAGRAM [*] Ruolo: contiene	
COMPRESO	Un class diagram comprende almeno un'associazione.	CLASS DIAGRAM [1..*] Ruolo: comprende ASSOCIAZIONE [1] Ruolo: è compresa	
DESCRIZIONE	Un'associazione potrebbe essere descritta tramite una classe di associazione.	ASSOCIAZIONE [0..1] Ruolo: è descritta CLASSE DI ASSOCIAZIONE [1] Ruolo: descrive	
POSSESSO	Una classe di associazione potrebbe possedere degli attributi.	CLASSE DI ASSOCIAZIONE [0..*] Ruolo: possiede ATTRIBUTO [0..1] Ruolo: è posseduto	
POSSESSO	Una classe potrebbe possedere degli attributi.	CLASSE [0..*] Ruolo: possiede ATTRIBUTO[0..1] Ruolo: è posseduto	
CONSISTE	Una classe potrebbe essere una classe di associazione.	CLASSE DI ASSOCIAZIONE [1] Ruolo: è CLASSE [0..1] Ruolo: può essere	
AGGREGAZIONE	Una classe potrebbe essere un'aggregazione di altre classi.	CLASSE [0..1] Ruolo: aggregata CLASSE [1..*] Ruolo: aggreganti	
COMPOSIZIONE	Una classe potrebbe essere una composizione di altre classi.	CLASSE [0..1] Ruolo: composta CLASSE[1..*] Ruolo: componenti	
GENERALIZZAZIONE	Una classe potrebbe essere una generalizzazione di altre classi.	CLASSE [0..1] Ruolo: generale CLASSE [1..*] Ruolo: specializzata	
PARTECIPA	Una o più associazioni connettono una o più classi.	ASSOCIAZIONE [1..*] Ruolo: connette CLASSE [1..*] Ruolo: è connessa	Ruoli: String Cardinalità: String Direzionalità: String

DIZIONARIO DELLE ASSOCIAZIONI

REALIZZAZIONE	Un tipo può essere realizzato da una classe.	TIPO [0..1] Ruolo: è realizzato CLASSE [1] Ruolo: realizza	
DOTAZIONE	Una classe potrebbe essere dotata di metodi.	METODO [1] Ruolo: è dote CLASSE [0..*] Ruolo: è dotata	
CARATTERIZZAZIONE	Un tipo potrebbe caratterizzare dei metodi.	METODO [0..1] Ruolo: ha TIPO [0..*] Ruolo: caratterizza	
CARATTERIZZAZIONE	Un tipo potrebbe caratterizzare un attributo.	TIPO [0..*] Ruolo: caratterizza ATTRIBUTO [0..1] Ruolo: ha	
COMPOSIZIONE	Un class diagram è composto da almeno una classe.	CLASS DIAGRAM[1..*] Ruolo: è composto CLASSE [1] Ruolo: compone	
COMPRESO	Un class diagram può avere da 0 a molte classi di associazione.	CLASS DIAGRAM[0..*] Ruolo: comprende CLASSE DI ASSOCIAZIONE [1] Ruolo : è compresa	
COMPRESO	Un class diagram può avere da 0 a molte enumeration.	ENUMERATION [1] Ruolo: è compresa CLASS DIAGRAM[0..*] Ruolo: comprende	
COMPONIMENTO	Uno o più valori, sono contenuti in una o più enumeration.	VALORE[1..*] Ruolo: è associato ENUMERATION[1..*] Ruolo: assume	N_Valori: integer
CONSISTENZA	Un tipo potrebbe essere un enumeration.	TIPO [0..1] Ruolo: può essere ENUMERATION[1] Ruolo: è	
CARATTERIZZAZIONE	Ogni parametro ha un suo tipo.	TIPO [0..*] Ruolo: Caratterizza PARAMETRO [1] Ruolo: è caratterizzato	
DOMINIO	Specifica I parametri di un metodo.	METODO [0..*] Ruolo: ha PARAMETRO [1..*] Ruolo: è posseduto	N_Parametri: integer

DIZIONARIO DEI VINCOLI

NOME VINCOLO	DESCRIZIONE
TipoP	Specifica che un tipo primitivo può assumere uno ed uno solo tra i seguenti valori: integer , char ,boolean , date , double, float, short, long, byte.
Mod1	Specifica che modifyacc di un attributo può assumere uno ed uno solo tra I seguenti valori: Public, Private, Package, Protected.
VincoloTipo	Specifica che TIPO può essere uno ed uno solo tra TipoPrimitivo o Tipoclass o Enumeration (OR esclusivo).
Unico_Classe	Specifica che in un class diagram ogni classe deve avere un nome univoco.
Unico_Classe_Di_Associazione	Specifica che in un class diagram ogni classe di associazione deve avere un nome univoco.
Unico_Attributo	Specifica che in una classe o in una classe di associazione o in una classe astratta non possono esserci due attributi con lo stesso nome.
Scelta_Classe	Una classe può essere una ed una soltanto tra: classe, classe di associazione, classe astratta. (OR ESCLUSIVO).
Vincolo_Specializzazione	Specifica che una classe non può essere una specializzazione di se stessa.
Vincolo_Componente	Specifica che una classe non può essere una componente di se stessa.
Vincolo_Aggregante	Specifica che una classe non può essere un'aggregante di se stessa.
Vincolo_Enum	Specifica che non possono esserci due enumeration con lo stesso nome nello stesso class diagram.
Mod2	Specifica che modifyacc di un metodo può assumere uno ed uno solo tra I seguenti valori: Public, Private, Package, Protected.

Appartenenza_Attributo	Specifica che un attributo può essere associato o ad una classe, o ad una classe di associazione o ad una classe astratta (OR esclusivo).
Vincolo_Data1	Specifica che la data di aggiornamento del class diagram deve maggiore o uguale della data di creazione.
Vincolo_Data2	Specifica che la data di modifica di un commento deve essere maggiore o uguale della data di creazione.
Vincolo_Data3	Specifica che la data di modifica della classe deve essere maggiore o uguale della data di creazione.
Vincolo_Data4	Specifica che la data di modifica di un attributo deve essere maggiore o uguale della data di creazione.
Unico_Metodo	Specifica che all'interno della stessa classe due metodi non possono avere la stessa Segnatura.
Pro_OR_Fun	Specifica se il metodo è una procedura (Tipo di ritorno settato a NULL) o una funzione (Tipo di ritorno settato a NOT NULL) .
P_IN_OR_P_OUT	Specifica che un paramentro può essere un paramentro di input o un parametro di output o un parametro sia di input che di output (OR esclusivo).
NO_Classe	Specifica che se una classe è una classe di associazione allora non può essere Generale, Aggregata e composta.

DEFINIZIONI DELLE TABELLE

CREATE TABLE CLASSDIAGRAM

```
(  
    Codice VARCHAR (30),  
    Nome VARCHAR2(30),  
    FileC VARCHAR2(30) NOT NULL,  
    DataCreazione DATE NOT NULL,  
    DataAggiornamento DATE ,  
    CONSTRAINT pkclassdiagram PRIMARY KEY (Codice),  
    CONSTRAINT Vincolo_Data1 CHECK (DataCreazione <= DataAggiornamento)  
);
```

CREATE TABLE COMMENTO

```
(  
    Descrizione VARCHAR2(200) NOT NULL,  
    CodiceC INTEGER ,  
    Codice VARCHAR2(30) NOT NULL,  
    DataInserimento DATE NOT NULL,  
    DataModifica DATE ,  
    CONSTRAINT pkcommento PRIMARY KEY (CodiceC),  
    CONSTRAINT Vincolo_Data2 CHECK (DataInserimento <= DataModifica),  
    CONSTRAINT fkcommento FOREIGN KEY (Codice) REFERENCES CLASSDIAGRAM (Codice) ON DELETE  
        CASCADE  
);
```

CREATE TABLE ASSOCIAZIONE

```
(  
    CodAss INTEGER,  
    NomeAss VARCHAR(30) ,
```

```
Grado INTEGER NOT NULL,  
Codice VARCHAR(30) NOT NULL,  
IdAss INTEGER ,  
CONSTRAINT pkassociazione PRIMARY KEY (CodAss),  
CONSTRAINT fkassociazione1 FOREIGN KEY (Codice) REFERENCES CLASSDIAGRAM (Codice),  
CONSTRAINT fkassociazione2 FOREIGN KEY (IdAss) REFERENCES CLASSE_DI_ASSOCIAZIONE (IdAss)  
);
```

CREATE TABLE CLASSE

```
(  
    IdC INTEGER ,  
    NomeC VARCHAR2(30) NOT NULL ,  
    Descrizione VARCHAR2(100),  
    Codice VARCHAR2(30) NOT NULL,  
    Generale INTEGER ,  
    Aggregata INTEGER ,  
    Composta INTEGER ,  
    IdAss INTEGER ,  
    TipoClasse VARCHAR2 (30),  
    DataInserimento DATE NOT NULL,  
    DataModifica DATE ,  
CONSTRAINT pkclasse PRIMARY KEY (IdC),  
CONSTRAINT Unico_Classe UNIQUE (NomeC, Codice),  
CONSTRAINT Scelta_Classe CHECK( ( IdAss IS NOT NULL AND TipoClasse IS NULL AND IdC=IdAss ) OR  
        ( IdAss IS NULL AND TipoClasse IS NOT NULL AND (TipoClasse ='classe_concreta' OR  
        TipoClasse='classe_astratta') )),  
CONSTRAINT NO_Classe CHECK (( Idc = IdAss AND Generale IS NULL AND Aggregata IS NULL AND  
        Composta IS NULL) OR (Idc <> IdAss) ),  
CONSTRAINT Vincolo_Specializzazione CHECK (IdC <> Generale),  
CONSTRAINT Vincolo_Componente CHECK (IdC <> Composta),  
CONSTRAINT Vincolo_Aggregante CHECK (IdC <> Aggregata),  
CONSTRAINT Vincolo_Data3 CHECK (DataInserimento <= DataModifica),
```

```

CONSTRAINT fkclasse1 FOREIGN KEY (Generale) REFERENCES CLASSE(IdC) ON DELETE SET NULL,
CONSTRAINT fkclasse2 FOREIGN KEY (Aggregata) REFERENCES CLASSE (IdC) ON DELETE SET NULL,
CONSTRAINT fkclasse3 FOREIGN KEY (Composta) REFERENCES CLASSE (IdC) ON DELETE CASCADE,
CONSTRAINT fkclasse4 FOREIGN KEY (Codice) REFERENCES CLASSDIAGRAM (Codice) ON DELETE
    CASCADE,
CONSTRAINT fkclasse5 FOREIGN KEY (IdAss) REFERENCES CLASSE_DI_ASSOCIAZIONE (IdAss) ON
    DELETE CASCADE
);


```

CREATE TABLE ATTRIBUTO

```

(
    CodiceA INTEGER,
    NomeAt VARCHAR2(30) NOT NULL,
    Cardinalità VARCHAR2(30) NOT NULL,
    ModifyAcc VARCHAR2(9) NOT NULL,
    Descrizione VARCHAR2(100),
    CodiceT  INTEGER ,
    IdC  INTEGER,
    IdAss  INTEGER ,
    DataInserimento DATE NOT NULL,
    DataModifica DATE ,
    CONSTRAINT pkatributo PRIMARY KEY (CodiceA),
    CONSTRAINT Unico_Attributo UNIQUE (NomeAt ,IdC, IdAss),
    CONSTRAINT Appartenenza_Attributo CHECK ((IdC = IdAss ) OR (IdC IS NOT NULL AND IdAss IS  NULL )),
    CONSTRAINT Mod1 CHECK ((ModifyAcc = 'Private' OR ModifyAcc = 'Public' OR ModifyAcc ='Package' OR
        ModifyAcc = 'Protected') OR (ModifyAcc = 'private' OR ModifyAcc= 'public' OR
        ModifyAcc = 'package' OR ModifyAcc = 'protected')),
    CONSTRAINT Vincolo_Data4 CHECK (DataInserimento <= DataModifica),
    CONSTRAINT fkatributo1 FOREIGN KEY (CodiceT) REFERENCES TIPO (CodiceT) ON DELETE CASCADE,
    CONSTRAINT fkatributo2 FOREIGN KEY (IdC) REFERENCES CLASSE (IdC) ON DELETE CASCADE,
    CONSTRAINT fkatributo3 FOREIGN KEY (IdAss) REFERENCES CLASSE_DI_ASSOCIAZIONE (IdAss) ON
        DELETE CASCADE
);


```

CREATE TABLE METODO

```
(  
    Segnatura VARCHAR2(30),  
    Nome  VARCHAR2(30) NOT NULL,  
    ModifyAcc VARCHAR2(9) NOT NULL,  
    Corpo  VARCHAR2(30) ,  
    CodiceT  INTEGER ,  
    IdC  INTEGER NOT NULL,  
    Procedura INTEGER NOT NULL,  
    Funzione  INTEGER NOT NULL,  
    CONSTRAINT pkmetodo PRIMARY KEY (Segnatura),  
    CONSTRAINT Mod2 CHECK ( (ModifyAcc='Private') OR ( ModifyAcc='Public' ) OR (ModifyAcc ='Package')  
        OR (ModifyAcc = 'Protected') OR (ModifyAcc = 'private') OR ( ModifyAcc = 'public' ) OR  
        ( ModifyAcc = 'package' ) OR (ModifyAcc = 'protected') ) ,  
    CONSTRAINT Pro_OR_Fun CHECK ((Procedura = 1 AND Funzione = 0 AND CodiceT IS NULL) OR  
        (Procedura = 0 AND Funzione = 1 AND CODICET IS NOT NULL)),  
    CONSTRAINT Unico_Metodo UNIQUE (Segnatura, IdC),  
    CONSTRAINT fkmetodo1 FOREIGN KEY (CodiceT) REFERENCES TIPO (CodiceT) ON DELETE CASCADE,  
    CONSTRAINT fkmetodo2 FOREIGN KEY (IdC) REFERENCES CLASSE (IdC) ON DELETE CASCADE  
);
```

CREATE TABLE TIPO

```
(  
    CodiceT INTEGER,  
    TipoPrimitivo VARCHAR2(7) ,  
    TipoClasse VARCHAR2(30),  
    IdC INTEGER ,  
    CodiceE INTEGER ,  
    CONSTRAINT pktipo PRIMARY KEY (CodiceT),
```

```

CONSTRAINT TipoP CHECK ( (TipoPrimitivo= 'integer') OR (TipoPrimitivo = 'char') OR (TipoPrimitivo = 'boolean') OR (TipoPrimitivo = 'float') OR (TipoPrimitivo = 'date') OR (TipoPrimitivo = 'double') OR (TipoPrimitivo = 'short') OR (TipoPrimitivo = 'long') OR (TipoPrimitivo = 'byte') ),

CONSTRAINT VincoloTipo CHECK( ((TipoPrimitivo IS NOT NULL) AND (TipoClasse IS NULL) AND (CodiceE IS NULL )) OR ( (TipoPrimitivo IS NULL) AND (TipoClasse IS NOT NULL) AND (CodiceE IS NULL)) OR ( (TipoPrimitivo IS NULL) AND (TipoClasse IS NULL) AND (CodiceE IS NOT NULL)) ),

CONSTRAINT fktipo1 FOREIGN KEY (IdC) REFERENCES CLASSE (IdC) ON DELETE CASCADE,
CONSTRAINT fktipo2 FOREIGN KEY (CodiceE) REFERENCES ENUMERATION (CodiceE) ON DELETE CASCADE

);


```

CREATE TABLE PARTECIPA

```

(
    CodAss INTEGER NOT NULL,
    Ruoli VARCHAR2(30),
    Cardinalità VARCHAR2(30) NOT NULL,
    IdC INTEGER NOT NULL,
    Direzionalità VARCHAR2 (14) NOT NULL
    CONSTRAINT fkpartecipa1 FOREIGN KEY (CodAss) REFERENCES ASSOCIAZIONE (CodAss),
    CONSTRAINT fkpartecipa2 FOREIGN KEY (IdC) REFERENCES CLASSE (IdC)
);


```

CREATE TABLE CLASSE_DI_ASSOCIAZIONE

```

(
    IdAss INTEGER ,
    NomeA VARCHAR2(30) NOT NULL,
    Codice VARCHAR2(30) NOT NULL,
    CONSTRAINT pkclasse_di_associazione PRIMARY KEY (IdAss),
    CONSTRAINT Unico_Classe_Di_Associazione UNIQUE (NomeA, Codice),
    CONSTRAINT fkclasse_di_associazione FOREIGN KEY (Codice) REFERENCES CLASSDIAGRAM (Codice) ON DELETE CASCADE
);


```

CREATE TABLE ENUMERATION

```
(  
    CodiceE INTEGER,  
    NomeE VARCHAR2(30) NOT NULL,  
    Codice VARCHAR2(30) NOT NULL,  
    CONSTRAINT pkenumeration PRIMARY KEY (CodiceE),  
    CONSTRAINT Vincolo_Enum UNIQUE (NomeE ,Codice),  
    CONSTRAINT fkenumeration FOREIGN KEY (Codice) REFERENCES CLASSDIAGRAM (Codice) ON DELETE  
    CASCADE  
);
```

CREATE TABLE VALORE

```
(  
    NomeV VARCHAR2(30),  
    CONSTRAINT pkvalore PRIMARY KEY (NomeV)  
);
```

CREATE TABLE PARAMETRO

```
(  
    CodiceP INTEGER,  
    NomeP VARCHAR2(30) NOT NULL,  
    CodiceT INTEGER NOT NULL,  
    P_IN INTEGER NOT NULL,  
    P_OUT INTEGER NOT NULL,  
    CONSTRAINT P_IN_OR_P_OUT CHECK ((P_IN = 1 OR P_IN = 0) AND ( P_OUT = 0 OR P_OUT = 1)),  
    CONSTRAINT pkparametro PRIMARY KEY (CodiceP),  
    CONSTRAINT fkparametro FOREIGN KEY (CodiceT) REFERENCES TIPO (CodiceT) ON DELETE CASCADE  
);
```

CREATE TABLE COMPORIMENTO

```
(  
    N_valori INTEGER NOT NULL,  
    NomeV VARCHAR2(30) NOT NULL,  
    CodiceE INTEGER NOT NULL,  
    CONSTRAINT fkcomponimento1 FOREIGN KEY (CodiceE) REFERENCES ENUMERATION (CodiceE),  
    CONSTRAINT fkcomponimento2 FOREIGN KEY (NomeV) REFERENCES VALORE (NomeV)  
);
```

CREATE TABLE DOMINIO

```
(  
    N_Parametri INTEGER NOT NULL,  
    Segnatura VARCHAR2(30) NOT NULL,  
    CodiceP INTEGER NOT NULL,  
    CONSTRAINT fk1dominio FOREIGN KEY (Segnatura) REFERENCES METODO (Segnatura) ,  
    CONSTRAINT fk2dominio FOREIGN KEY (CodiceP) REFERENCES PARAMETRO (CodiceP)  
);
```