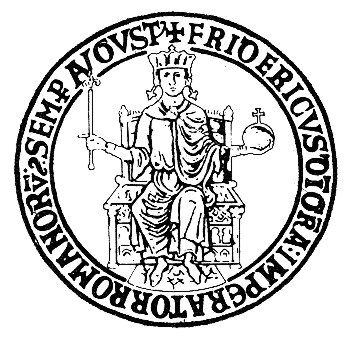
**UNIVERSITÀ DEGLI STUDI DI NAPOLI**

**“FEDERICO II”**

**DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL’INFORMAZIONE**



CORSO DI LAUREA IN INFORMATICA

**Progettazione e sviluppo di un database per la descrizione e memorizzazione di Class Diagram UML, con supporto a tutte le relative caratteristiche**

**DOCENTI:**

Prof. Adriano Peron

Prof. Alessandro De Luca

**AUTORI:**

Sara Esposito **MATR.**N87/1089

Luca Esposito **MATR.**N86/2565

**ANNO ACCADEMICO 2018-2019**

Indice

**I – Analisi del problema1**

**II – Progettazione concettuale2**

Class Diagram non ristrutturato2

Class Diagram ristrutturato3

**III – Dizionario dei dati4**

Dizionario delle classi4

Dizionario delle associazioni6

Dizionario dei vincoli………………………………………………………………………………………..7

**IV – Schema logico7**

**V – Progettazione fisica10**

Definizione delle tabelle10

Tabella 10

Tabella 10

Tabella 10

Tabella 11

Tabella 11

Tabella 11

Tabella 12

Tabella 12

Tabella 13

Tabella 13

Ulteriori chiavi esterne13

**I – Analisi del problema**

Il database che si andrà a progettare e successivamente ad implementare avrà il compito di rappresentare attraverso una base di dati un class diagram UML riferito ad un linguaggio di programmazione ad oggetti generico, con tutte le sue principali caratteristiche e funzionalità.

Il seguente database prevede di memorizzare le informazioni specifiche ad un relativo class diagram, come la gestione di classi con i rispettivi attributi, associazioni, tipi e metodi.

Si presuppone di voler permettere la memorizzazione delle classi, che possono essere anche classi di associazione o specializzazioni di altre classi, gestendo le associazioni (*ricorsive, singole e multiple*) tra di esse ed eventualmente i casi specifici come le aggregazioni e le composizioni.

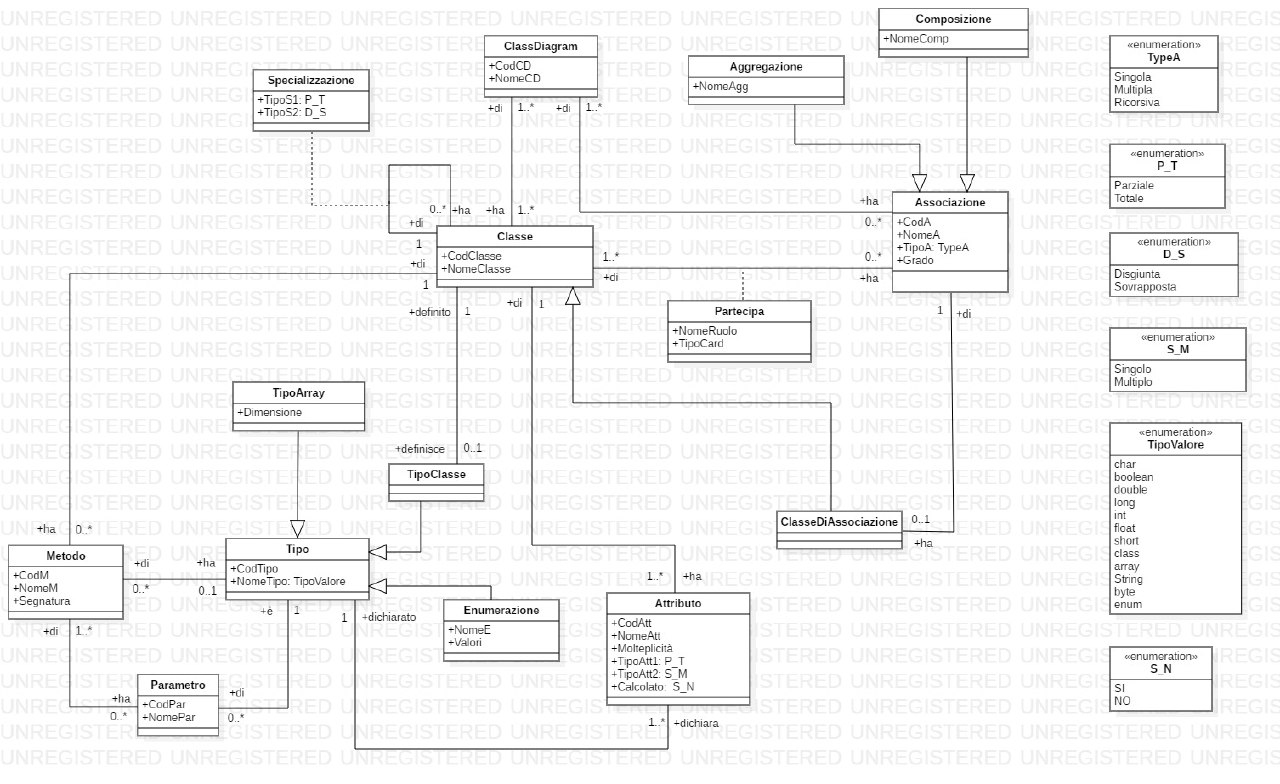
Gli attributi delle classi dovranno poter poi essere associati ai rispettivi tipi, distinguendo i tipi primitivi dai tipi complessi (*class, array, enum*).

**II – Progettazione Concettuale**

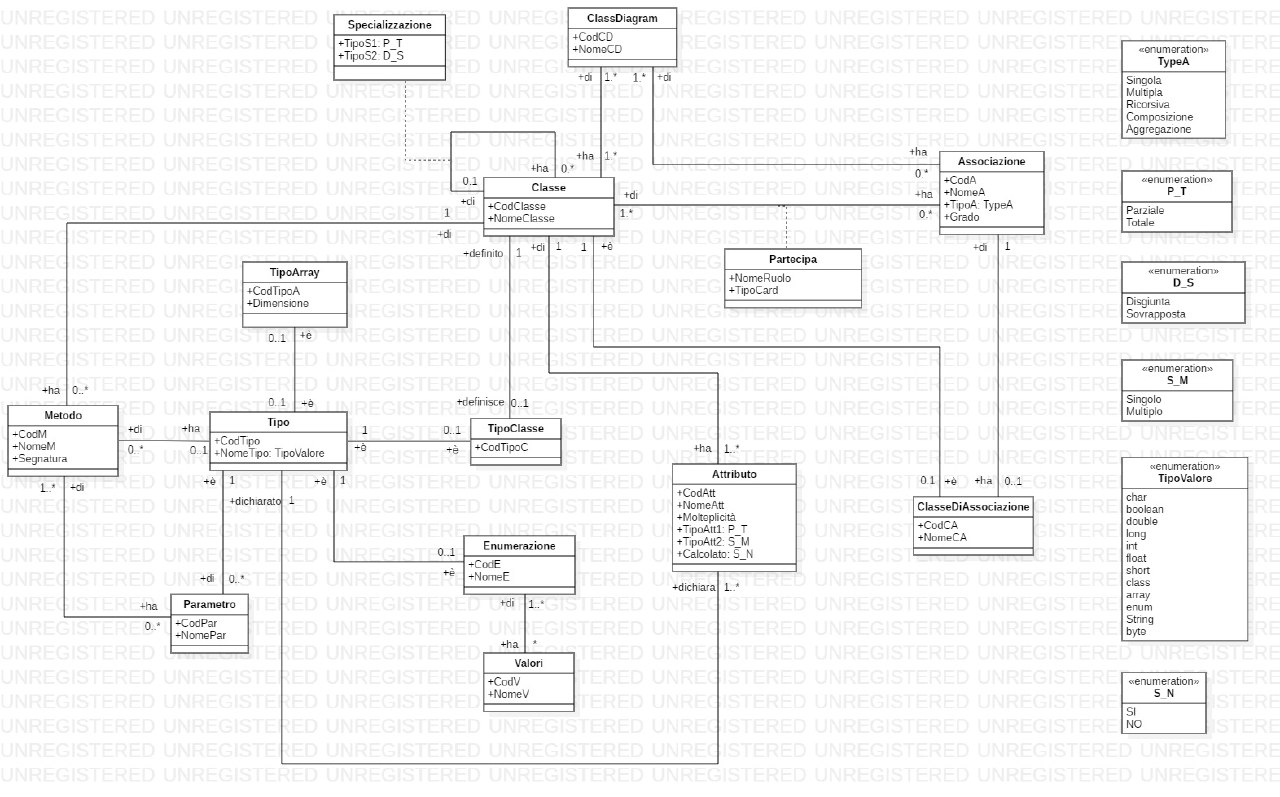
In questa fase formalizziamo i dati raccolti e analizzati durante l’analisi del problema, senza prendere in considerazione problematiche quali la struttura dati più adatta all’implementazione finale, possibili ridondanze tra i dati ecc.

Il class diagram così ottenuto, sarà successivamente ristrutturato in modo da rendere più efficiente la sua implementazione. Gli attributi multipli, gli attributi strutturati e le specializzazioni vanno rimosse e sostituite con classi o associazioni che ne simulano il comportamento.

**Class diagram non ristrutturato:**



**Class diagram ristrutturato:**

****

Le modifiche adottate per ottenere tale class diagram sono state:

- sostituire le generalizzazioni delle classi TipoArray, TipoClasse, Enumerazione e ClasseDiAssociazione con delle associazioni, distinguendo le varie classi con dei nuovi attributi (CodTipoA per TipoArray, CodTipoC per TipoClasse, CodE, NomeE per Enumerazione e CodCA, NomeCA per ClasseDiAssociazione) ;

- poiché le sottoclassi Composizione e Aggregazione sono delle associazioni particolari, possono essere considerate come istanze della classe Associazione, distinguendole attraverso due nuovi valori nell’enumerazione TypeA (Composizione e Aggregazione);

- infine l’attributo multiplo Valori della sottoclasse Enumerazione è stato sostituito con una nuova classe Valori che ha come attributi CodV e NomeV.

**III – Dizionari**

**Dizionario delle classi:**

Il dizionario delle classi permette di descrivere ciò che ogni classe e attributo andrà a rappresentare all’interno del class diagram ristrutturato.

|  |  |  |  |
| --- | --- | --- | --- |
| NOME CLASSE | DESCRIZIONE | ATTRIBUTI | NOTE |
| **ClassDiagram** | Descrive l’insieme delle classi con le rispettive caratteristiche e le eventuali associazioni. | **CodCD**:  - VARCHAR(5)  - molteplicità: [1]  - codice del class diagram  - attributo identificativo  **NomeCD**:  -VARCHAR(20)  - molteplicità: [1]  - nome del class diagram  - attributo totale, singolo | / |
| **Classe** | Descrive le classi contenute in un class diagram. | **CodClasse**:  - VARCHAR(5)  - molteplicità: [1]  - codice della classe  - attributo identificativo  **NomeClasse**:  - VARCHAR(20)  - molteplicità: [1]  - nome della classe  - attributo totale, singolo | / |
| **ClasseDiAssociazione** | Descrive le classi associate ad una relazione tra classi. | **CodCA**:  - VARCHAR(5)  - molteplicità: [1]  - codice della classe di associazione  - attributo identificativo  **NomeCA**:  - VARCHAR(20)  - molteplicità: [1]  - nome della classe di associazione  - attributo totale,singolo | / |
| **Associazione** | Descrive le associazioni delle varie classi relazionate. | **CodA**:  - VARCHAR(5)  - molteplicità: [1]  - codice dell’associazione  - attributo identificativo  **NomeA**:  - VARCHAR(20)  - molteplicità: [1]  - nome dell’associazione  - attributo totale, singolo  **TipoA**:  - TypeA (ENUM)  - molteplicità: [1]  - tipo dell’associazione  - attributo totale, singolo  **Grado**:  - INT  - molteplicità: [1]  - numero delle entità coinvolte  - attributo totale, calcolato | **TipoA** può assumere i valori:  - Singola  - Multipla  - Rircorsiva  - Composizione  - Aggregazione |
| **Attributo** | Descrive gli attributi associati alle classi. | **CodAtt**:  - VARCHAR(5)  - molteplicità: [1]  - codice dell’attributo  - attributo identificativo  **NomeAtt**:  - VARCHAR(20)  - molteplicità: [1]  - nome dell’attributo  - attributo totale, singolo  **Molteplicità**:  - VARCHAR(20)  -molteplicità: [1]  - molteplicità dell’attributo  - attributo totale, singolo  **TipoAtt1**:  - P\_T (ENUM)  - molteplicità: [1]  - tipo dell’attributo  - attributo totale, singolo  **TipoAtt2**:  - S\_M (ENUM)  - molteplicità: [1]  - tipo dell’attributo  - attributo totale, singolo  **Calcolato**:  - S\_N (ENUM)  - molteplicità: [1]  - indica se un attributo è calcolato  - attributo totale, singolo | **TipoAtt1** può assumere i valori:  - Parziale  - Totale  **TipoAtt2** può assumere i valori:  - Singola  - Multipla  **Calcolato** può assumere i valori:  - SI  - NO |
| **Tipo** | Descrive il tipo di valore che può assumere un dato. | **CodTipo**:  - VARCHAR(5)  - molteplicità: [1]  - codice del tipo  - attributo identificativo  **NomeTipo**:  - TipoValore (ENUM)  - molteplicità: [1]  - nome del tipo  - attributo totale, singolo | **NomeTipo** può assumere i valori:  - char  - boolean  - double  - long  - int  - float  - short  - class  - array  - enum  - String  - byte |
| **TipoClasse** | Descrive il tipo di dato che può assumere come valore i campi di una classe. | **CodTipoC**  - VARCHAR(5)  - molteplicità: [1]  - codice del tipo classe  - attributo identificativo | / |
| **TipoArray** | Descrive il tipo di dato che può assumere come valore un vettore di elementi. | **CodTipoA**  - VARCHAR(5)  - molteplicità: [1]  - codice del tipo array  - attributo identificativo  **Dimensione**:  - INT  - molteplicità: [1]  - numero di elementi contenuti dall’array  - attributo totale, singolo | / |
| **Enumerazione** | Descrive un elenco di valori specifici che può assumere un dato. | **CodE**:  - VARCHAR(5)  - molteplicità: [1]  - codice dell’enumerazione  - attributo identificativo  **NomeE**:  - VARCHAR(20)  - molteplicità: [1]  - nome dell’enumerazione  - attributo totale, singolo | / |
| **Valori** | Descrive i valori di una enumerazione specifica. | **CodV**  -VARCHAR(5)  -molteplicità: [1]  - codice dei valori  -attributo identificativo  **NomeV:**  - VARCHAR(20)  - molteplicità: [1]  - nome dei valori  - attributo totale, singolo | / |
| **Metodo** | Descrive i metodi contenuti nelle classi. | **CodM**:  - VARCHAR(5)  - molteplicità: [1]  - codice del metodo  - attributo identificativo  **NomeM**:  - VARCHAR(20)  - molteplicità: [1]  - nome del Metodo  - attributo totale, singolo  **Segnatura**:  - VARCHAR(500)  - molteplicità: [1]  - segnatura del metodo  - attributo totale, singolo | / |
| **Parametro** | Descrive i parametri utilizzati dai vari metodi. | **CodPar**:  - VARCHAR(5)  - molteplicità: [1]  - codice del parametro  - attributo identificativo  **NomePar**:  - VARCHAR(20)  - molteplicità: [1]  - nome del parametro  - attributo totale, singolo | / |

**Dizionario delle associazioni:**

Il dizionario delle associazioni descrive come sono relazionate le varie classi all’interno della base di dati.

|  |  |  |  |
| --- | --- | --- | --- |
| NOME ASSOCIAZIONE | DESCRIZIONE | CLASSI COINVOLTE | CLASSI DI ASSOCIAZIONE |
| **Contiene Classi** | Indica le classi che compongono un class diagram *(associazione singola)*. | **ClassDiagram**  - *ruolo*: ha [1.\*]  - un class diagram ha delle classi  **Classe**  - *ruolo*: di [1.\*]  - la classe che appartiene al class diagram | / |
| **Contiene Associazioni** | Indica le associazioni che compongono un class diagram *(associazione singola)*. | **ClassDiagram**  - *ruolo*: ha [0.\*]  - un class diagram può avere delle associazioni  **Associazione**  - *ruolo*: di [1.\*]  - le associazioni che appartengono al class diagram | / |
| **Specializzazione** | Indica le specializzazioni che può avere una classe *(associazione ricorsiva)*. | **Classe**  - *ruolo*: ha [0.\*]  - una classe ha delle specializzazioni  **Classe**  - *ruolo*: di [0.1]  - una specializzazione di una classe | **TipoS1**:  - P\_T (ENUM)  *(‘Parziale’,’Totale’)*  - molteplicità: [1]  - tipo della specializzazione  - attributo totale, singolo  **TipoS2**:  - D\_S (ENUM)  *(‘Disgiunta’,’Sovrapposta’)*  - molteplicità: [1]  - tipo della specializzazione  - attributo totale, singolo |
| **SpecCA** | Indica un tipo particolare di classe, le classi di associazione  *(associazione singola)*. | **Classe**  - *ruolo*: è [0.1]  - una classe può essere una classe di associazione  **ClassedDiAssociazione**  - *ruolo*: è [1]  - una classe di associazione è una classe | / |
| **Partecipa** | Indica le classi che sono relazionate da una o più associazioni *(associazione singola)*. | **Classe**  - *ruolo*: ha [0.\*]  - una classe ha delle associazioni  **Associazione**  - *ruolo*: di [1.\*]  - l’associazione delle classi coinvolte | **NomeRuolo**:  - VARCHAR(20)  - molteplicità: [0.\*]  - nome del ruolo  - attributo parziale, multiplo  **TipoCard**:  - VARCHAR(20)  - molteplicità: [2.\*]  - tipo della cardinalità  - attributo totale, multiplo |
| **Correlata** | Indica l’associazione tra una classe e l’associazione a cui è correlata *(associazione singola)*. | **Associazione**  - *ruolo*: ha [0.1]  - un’associazione ha una classe di associazione  **ClasseDiAssociazione**  - *ruolo*: di [1]  - una classe di associazione di un’associazione | / |
| **Contiene Attributo** | Indica gli attributi associati ad una classe *(associazione singola)*. | **Attributo**  - *ruolo*: di [1]  - un attributo di una classe  **Classe**  - *ruolo*: ha [1.\*]  - una classe ha uno o più attributi | / |
| **SpecEnum** | Indica un particolare tipo di dato, le enumerazioni  *(associazione singola)*. | **Tipo**  - *ruolo*: è [0.1]  - un tipo può essere un’enumerazione  **Enumerazione**  - *ruolo*: è [1]  - un’enumerazione è un tipo | / |
| **SpecTipoC** | Indica un particolare tipo di dato, i tipi di classe  *(associazione singola)*. | **Tipo**  - *ruolo*: è [0.1]  - un tipo può essere di tipo classe  **TipoClasse**  - *ruolo*: è [1]  - tipo di classe è un tipo | / |
| **SpecTipoA** | Indica un particolare tipo di dato, gli array  *(associazione singola)*. | **Tipo**  - *ruolo*: è [0.1]  - un tipo può essere un array di elementi  **TipoArray**  - *ruolo*: è [1]  - tipo array è un tipo | / |
| **Dichiara Attributo** | Indica il tipo che può assumere un attributo *(associazione singola)*. | **Attributo**  - *ruolo*: dichiarato [1]  - un attributo è dicharato  **Tipo**  - *ruolo*: dichiara [1.\*]  - un tipo dichiara uno o più attributi | / |
| **Dichiara Tipo di Classe** | Definisce un tipo come un tipo di classe  *(associazione singola)*. | **Classe**  - *ruolo*: definisce [0.1]  - la classe definisce un tipo classe  **TipoClasse**  - *ruolo*: definito [1]  - il tipo è definito di tipo classe | / |
| **Dichiara Parametro** | Indica il tipo che può assumere un parametro *(associazione singola)*. | **Tipo**  - *ruolo*: di [0.\*]  - tipo di uno o più parametri  **Parametro**  - *ruolo*: è [1]  - un parametro è di un certo tipo | / |
| **Dichiara ritorno Metodo** | Indica il tipo di ritorno che può avere un metodo *(associazione singola)*. | **Tipo**  - *ruolo*: di [0.\*]  - tipo di ritorno di uno o più metodi  **Metodo**  - *ruolo*: ha [0.1]  - tipo di ritorno di un metodo | / |
| **Contiene Metodo** | Indica i metodi contenuti in una classe *(associazione singola)*. | **Metodo**  - *ruolo*: di [1]  - un metodo è di una classe  **Classe**  - *ruolo*: ha [0.\*]  - una classe può avere uno o più metodi | / |
| **Contiene Parametro** | Indica i parametri contenuti nei metodi *(associazione singola)*. | **Metodo**  - *ruolo*: ha [0.\*]  - un metodo ha dei parametri  **Parametro**  - *ruolo*: di [1.\*]  -un parametro di un metodo | / |
| **Contiene Valori** | Indica i valori che contiene un’enumerazione *(associazione singola)*. | **Enumerazione**  - ruolo: ha [\*]  - un’enumerazione ha dei valori  **Valori**  - ruolo: di [1.\*]  - i valori di un’enumerazione | / |

**Dizionario dei vincoli:**

Il dizionario dei vincoli descrive i vincoli presenti all’interno della base di dati.

|  |  |
| --- | --- |
| NOME VINCOLO | DESCRIZIONE |
| **id\_nomeCD** | Non possono esistere due class diagram con lo stesso nome. |
| **id\_nomeClasse** | Non possono esistere più di una classe con lo stesso nome all’interno dello stesso class diagram. |
| **id\_nomeAssociazione** | Non possono esistere più di una associazione con lo stesso nome all’interno dello stesso class diagram. |
| **check\_grado\_tipoa** | Se Grado in Associazione è: >2 TipoA è ‘Multipla’,  =2 TipoAè ‘Singola’,  =1 TipoAè ‘Ricorsiva’ |
| **id\_componenti** | Non possono esistere classi componenti che sono associate a più di una classe composta. |
| **constraint\_ruolocomp** | Se TipoA in Associazione è ‘Composizione’, allora NomeRuolo in Partecipa può essere solo ‘composto’ o ‘componente’. |
| **constraint\_ruoloaggr** | Se TipoA in Associazione è ‘Aggregazione’, allora NomeRuolo in Partecipa può essere solo ‘aggregato’ o ‘aggregante’. |
| **id\_nomeCA** | Non possono esistere più di una classe di associazione con lo stesso nome. |
| **id\_nomeAttributo** | Non possono esistere due attributi con lo stesso nome associati alla stessa classe. |
| **check\_molteplicità** | Se TipoAtt1 in Attributo è ‘Totale’, allora Molteplicità deve essere diversa da 0. |
| **constraint\_nomeTipo** | NomeTipo in Tipo non può assumere i valori ‘array’, ‘class’ ed ‘enum’ se non esistono le rispettive istanze di TipoArray, TipoClasse e Enumerazione. |
| **constraint\_tipoArray** | Quando viene inserita un’istanza di TipoArray, si attiva un trigger che inserisce la rispettiva istanza di Tipo con NomeTipo=’array’. |
| **constraint\_tipoClasse** | Quando viene inserita un’istanza di TipoClasse, si attiva un trigger che inserisce la rispettiva istanza di Tipo con NomeTipo=’class’. |
| **constraint\_enumerazione** | Quando viene inserita un’istanza di Enumerazione, si attiva un trigger che inserisce la rispettiva istanza di Tipo con NomeTipo=’enum’. |
| **id\_nomeE** | Non possono esistere due enumerazioni con lo stesso nome. |
| **id\_nomeMetodo** | Non possono esistere due metodi con lo stesso nome associati alla stessa classe. |
| **id\_nomeParametro** | Non possono esistere due parametri con lo stesso nome associati allo stesso metodo. |

**IV – Schema logico**

In questa fase è rappresentato lo schema logico che definisce in maniera chiara ed efficiente le tabelle e le relazioni tra di esse, attraverso la rappresentazione esplicita delle chiavi esterne.

Nello schema seguente, le chiavi primarie vengono identificate da una singola sottolineatura mentre le chiavi esterne da una doppia sottolineatura.

**CLASSDIAGRAM** (CodCD, NomeCD)

**CLASSE** (CodClasse, NomeClasse)

**COMP\_CD\_C** (CodCD, CodClasse)

**SPECIALIZZAZIONE** (TipoS1, TipoS2, CodClasse, CodClasseS)

**ASSOCIAZIONE** (CodA, NomeA, TipoA, Grado)

**COMP\_CD\_A** (CodCD, CodA)

**PARTECIPA** (NomeRuolo, TipoCard, CodA, CodClasse)

**CLASSEDIASSOCIAZIONE** (CodCA, NomeCA, CodA , CodClasse)

**TIPO** (CodTipo, NomeTipo)

**ATTRIBUTO** (CodAtt, NomeAtt, Molteplicità, TipoAtt1, TipoAtt2, Calcolato, CodTipo, CodClasse)

**TIPOCLASSE** (CodTipoC, CodTipo, CodClasse)

**TIPOARRAY** (CodTipoA, Dimensione, CodTipo)

**ENUMERAZIONE** (CodE, NomeE, CodTipo)

**VALORI** (CodV, NomeV)

**COMP\_VAL** (CodV, CodE)

**PARAMETRO** (CodPar, NomePar, TipoPar)

**METODO** (CodM, NomeM, Segnatura, CodClasse)

**CMP\_PAR\_M** (CodM, CodPar)

**V – Progettazione fisica**

Per la gestione della base di dati è stato usato il database management system **Oracle XE 12c**.

**Definizione delle tabelle:**

*/\*\**

*\* TABELLA: ClassDiagram*

*\* Crea la tabella ClassDiagram e implementa i vincoli intrarelazionali*

*\*/*

CREATE TABLE ClassDiagram

(CodCD VARCHAR(5),

NomeCD VARCHAR(20) NOT NULL,

*-- Vincoli*

CONSTRAINT pk\_codCD PRIMARY KEY (CodCD),

CONSTRAINT id\_nomeCD UNIQUE(NomeCD)

);

/

*/\*\**

*\* TABELLA: Classe*

*\* Crea la tabella Classe e implementa i vincoli intrarelazionali*

*\*/*

CREATE TABLE Classe

(CodClasse VARCHAR(5),

NomeClasse VARCHAR(20) NOT NULL,

*-- Vincoli*

CONSTRAINT pk\_codC PRIMARY KEY (CodClasse)

);

/

*/\*\**

*\* TABELLA: Comp\_CD\_C*

*\* Crea la tabella Comp\_CD\_C e implementa i vincoli interrelazionali*

*\*/*

CREATE TABLE Comp\_CD\_C

(CodClasse VARCHAR(5),

CodCD VARCHAR(5),

*-- Vincoli*

CONSTRAINT fk\_comp\_cdc\_codC FOREIGN KEY (CodClasse) REFERENCES Classe (CodClasse),

CONSTRAINT fk\_comp\_cdc\_codCD FOREIGN KEY (CodCD) REFERENCES ClassDiagram (CodCD)

);

/

*/\*\**

*\* TABELLA: Specializzazione*

*\* Crea la tabella Specializzazione e implementa i vincoli interrelazionali*

*\*/*

CREATE TABLE Specializzazione

(TipoS1 VARCHAR(8) CHECK (TipoS1 IN (‘Parziale’, ‘Totale’)),

TipoS2 VARCHAR(11) CHECK (TipoS2 IN (‘Disgiunta’, ‘Sovrapposta’),

CodClasse VARCHAR(5),

CodClasseS VARCHAR(5),

*-- Vincoli*

CONSTRAINT fk\_sp\_codC FOREIGN KEY (CodClasse) REFERENCES Classe (CodClasse),

CONSTRAINT fk\_sp\_codCS FOREIGN KEY (CodClasseS) REFERENCES Classe (CodClasse)

);

/

*/\*\**

*\* TABELLA: Associazione*

*\* Crea la tabella Associazionee implementa i vincoli intrarelazionali/ennupla*

*\*/*

CREATE TABLE Associazione

(CodA VARCHAR(5),

NomeA VARCHAR(20),

TipoA VARCHAR(9) CHECK (TipoA IN (‘Singola’, ‘Multipla’, ‘Ricorsiva’, ‘Composizione’, ‘Aggregazione’)),

Grado INT NOT NULL,

*-- Vincoli*

CONSTRAINT pk\_codA PRIMARY KEY (CodA),

CONSTRAINT check\_grado\_tipoa CHECK ((Grado=1 AND TipoA=’Ricorsiva’) OR (Grado=2 AND TipoA=’Singola’) OR (Grado>2 AND TipoA=’Multipla’))

);

/

*/\*\**

*\* TABELLA: Comp\_CD\_A*

*\* Crea la tabella Comp\_CD\_A e implementa i vincoli interrelazionali*

*\*/*

CREATE TABLE Comp\_CD\_A

(CodCD VARCHAR(5),

CodA VARCHAR(5),

*-- Vincoli*

CONSTRAINT fk\_comp\_cda\_codCD FOREIGN KEY (CodCD) REFERENCES ClassDiagram (CodCD),

CONSTRAINT fk\_comp\_cda\_codA FOREIGN KEY (CodCD) REFERENCES Associazione (CodA)

);

/

*/\*\**

*\* TABELLA: Partecipa*

*\* Crea la tabella Partecipa e implementa i vincoli interrelazionali*

*\*/*

CREATE TABLE Partecipa

(NomeRuolo VARCHAR(20),

TipoCard VARCHAR(20) NOT NULL,

CodClasse VARCHAR(5),

CodA VARCHAR(5),

*-- Vincoli*

CONSTRAINT fk\_par\_codC FOREIGN KEY (CodClasse) REFERENCES Classe (CodClasse),

CONSTRAINT fk\_ par\_codA FOREIGN KEY (CodA) REFERENCES Associazione (CodA)

);

/

*/\*\**

*\* TABELLA: ClasseDiAssociazione*

*\* Crea la tabella ClasseDiAssociazione e implementa i vincoli intrarelazionali/interrelazionali*

*\*/*

CREATE TABLE ClasseDiAssociazione

(CodCA VARCHAR(5),

NomeCA VARCHAR(20) NOT NULL,

CodClasse VARCHAR(5),

CodA VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codCA PRIMARY KEY (CodCA),

CONSTRAINT fk\_ca\_codC FOREIGN KEY (CodClasse) REFERENCES Classe (CodClasse),

CONSTRAINT fk\_ca\_codA FOREIGN KEY (CodA) REFERENCES Associazione (CodA),

CONSTRAINT id\_nomeCA UNIQUE(NomeCA)

);

/

*/\*\**

*\* TABELLA: Tipo*

*\* Crea la tabella Tipo e implementa i vincoli intrarelazionali/ennupla*

*\*/*

CREATE TABLE Tipo

(CodTipo VARCHAR(5),

NomeTipo VARCHAR(6) CHECK (NomeTipoIN (‘char’, ‘boolean’, double’, ‘long’, ‘int’, ‘float’, ‘short’, ‘class’, ‘array’, ‘enum’, ‘String’, ‘byte’)),

*-- Vincoli*

CONSTRAINT pk\_codT PRIMARY KEY (CodTipo)

);

/

*/\*\**

*\* TABELLA: Attributo*

*\* Crea la tabella Attributo e implementa i vincoli intrarelazionali/interrelazionali/ennupla*

*\*/*

CREATE TABLE Attributo

(CodAtt VARCHAR(5),

NomeAtt VARCHAR(20) NOT NULL,

Molteplicità VARCHAR(10) NOT NULL,

TipoAtt1 VARCHAR(8) CHECK (TipoAtt1 IN (‘Parziale’, ‘Totale’)),

TipoAtt2 VARCHAR(8) CHECK (TipoAtt2 IN (‘Singolo’, ‘Multiplo’)),

Calcolato CHAR CHECK (Calcolato IN (‘SI’, ‘NO’)),

CodTipo VARCHAR(5),

CodClasse VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codAtt PRIMARY KEY (CodAtt),

CONSTRAINT fk\_tipoa\_codT FOREIGN KEY (CodTipo) REFERENCES Tipo (CodTipo),

CONSTRAINT fk\_att\_codC FOREIGN KEY (CodClasse) REFERENCES Classe (CodClasse),

CONSTRAINT id\_nomeAtt UNIQUE (NomeAtt,CodClasse),

CONSTRAINT check\_molteplicità CHECK (TipoAtt1=’Totale’ AND Molteplicità<>’0’)

);

/

*/\*\**

*\* TABELLA: TipoClasse*

*\* Crea la tabella TipoClasse e implementa i vincoli intrarelazionali/interrelazionali*

*\*/*

CREATE TABLE TipoClasse

(CodTipoC VARCHAR(5),

CodTipo VARCHAR(5),

CodClasse VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codTC PRIMARY KEY (CodTipoC),

CONSTRAINT fk\_tipoc\_codT FOREIGN KEY (CodTipo) REFERENCES Tipo (CodTipo),

CONSTRAINT fk\_tipoc\_codC FOREIGN KEY (CodClasse) REFERENCES Classe(CodClasse)

);

/

*/\*\**

*\* TABELLA: TipoArray*

*\* Crea la tabella TipoArray e implementa i vincoli intrarelazionali/interrelazionali*

*\*/*

CREATE TABLE TipoArray

(CodTipoA VARCHAR(5),

Dimensione INT NOT NULL,

CodTipo VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codTA PRIMARY KEY (CodTipoA),

CONSTRAINT fk\_tipoa\_codT FOREIGN KEY (CodTipo) REFERENCES Tipo (CodTipo)

);

/

*/\*\**

*\* TABELLA: Enumerazione*

*\* Crea la tabella Enumerazione e implementa i vincoli intrarelazionali/interrelazionali*

*\*/*

CREATE TABLE Enumerazione

(CodE VARCHAR(5),

NomeE VARCHAR(20) NOTNULL,

CodTipo VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codE PRIMARY KEY (CodE),

CONSTRAINT fk\_enum\_codtipo FOREIGN KEY (CodTipo) REFERENCES Tipo (CodTipo),

CONSTRAINT id\_nomeE UNIQUE(NomeE)

);

/

*/\*\**

*\* TABELLA: Valori*

*\* Crea la tabella Valori e implementa i vincoli intrarelazionali*

*\*/*

CREATE TABLE Valori

(CodV VARCHAR(5),

NomeV VARCHAR(20) NOT NULL,

*-- Vincoli*

CONSTRAINT pk\_codV PRIMARY KEY (CodV),

);

/

*/\*\**

*\* TABELLA: Comp\_Val*

*\* Crea la tabella Comp\_Val e implementa i vincoli interrelazionali*

*\*/*

CREATE TABLE Comp\_Val

(CodV VARCHAR(5),

CodE VARCHAR(5),

*-- Vincoli*

CONSTRAINT fk\_codV FOREIGN KEY (CodV) REFERENCES Valori (CodV),

CONSTRAINT fk\_codE FOREIGN KEY (CodE) REFERENCES Enumerazione (CodE)

);

/

*/\*\**

*\* TABELLA: Parametro*

*\* Crea la tabella Parametro e implementa i vincoli intrarelazionali/interrelazionali*

*\*/*

CREATE TABLE Parametro

(CodPar VARCHAR(5),

NomePar VARCHAR(20) NOT NULL,

TipoPar VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codP PRIMARY KEY (CodPar),

CONSTRAINT fk\_par\_codT FOREIGN KEY (TipoPar) REFERENCES Tipo (CodTipo)

);

/

*/\*\**

*\* TABELLA: Metodo*

*\* Crea la tabella Metodo e implementa i vincoli intrarelazionali/interrelazionali*

*\*/*

CREATE TABLE Metodo

(CodM VARCHAR(5),

NomeM VARCHAR(20) NOT NULL,

Segnatura VARCHAR(500)NOT NULL,

CodClasse VARCHAR(5),

*-- Vincoli*

CONSTRAINT pk\_codM PRIMARY KEY (CodM),

CONSTRAINT fk\_m\_codC FOREIGN KEY (CodClasse) REFERENCES Classe (CodClasse),

CONSTRAINT id\_nomeMetodo UNIQUE(NomeM, CodClasse)

);

/

*/\*\**

*\* TABELLA: Comp\_Par\_M*

*\* Crea la tabellaComp\_Par\_M e implementa i vincoli interrelazionali*

*\*/*

CREATE TABLE Comp\_Par\_M

(CodM VARCHAR(5),

CodPar VARCHAR(5),

*-- Vincoli*

CONSTRAINT fk\_codM FOREIGN KEY (CodM) REFERENCES Metodo (CodM),

CONSTRAINT fk\_codP FOREIGN KEY (CodPar) REFERENCES Parametro (CodPar)

);