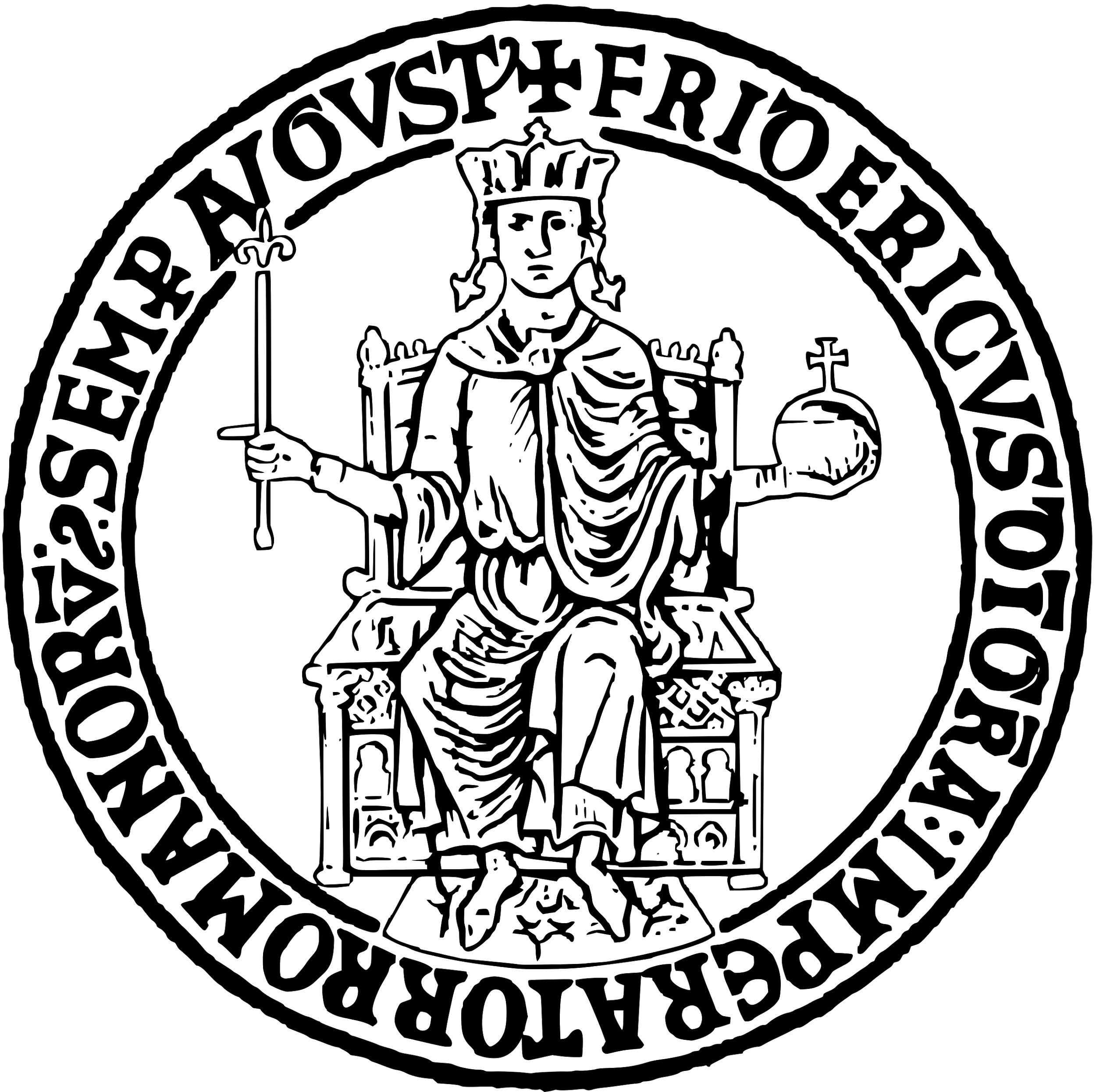
Università degli studi di Napoli Federico II  
Scuola Politecnica e delle Scienze di Base Dipartimento di Ingegneria Elettrica e Tecnologie dell’Informazione  
anno accademico 2020/2021

Progettazione e sviluppo di una base di dati relazionale per la gestione di progetti.  
  
Esposito Sara Trinchillo Giusi Ornella Beniamino Gentile

N86003374 N86003950 N86003004

Indice

1. Descrizione del progetto
2. Progettazione concettuale
3. Analisi di un class diagram
4. Class Diagram
5. Alcune precisazioni sul class diagram
6. Ristrutturazione del class diagram
7. Chiavi primarie
8. Attributi multipli
9. Attributi strutturati
10. Attributi derivati
11. Gerarchie di specializzazione
12. Dizionari dei dati
13. Dizionario delle classi
14. Dizionario delle associazioni
15. Dizionario dei vincoli
16. Progettazione Logica

**Capitolo 1**

**Descrizione del progetto**

**1.1 Analisi del problema**

Si progetti ed implementi una base di dati relazionale, ovvero uno schema concettuale di dati strutturati, che possa essere d’aiuto alla gestione e memorizzazione di progetti realizzati da un’azienda.   
Bisogna tener traccia dei dipendenti della suddetta azienda ricordando che ognuno di essi ha un ruolo.  
In ogni progetto ci sarà un unico project manager, avrà una specifica tipologia, che può essere: “Ricerca di base”, “Ricerca Industriale”, “Ricerca sperimentale”, “Sviluppo Sperimentale”, ed uno o più ambiti, tra cui: Economia, Medicina, Informatica.  
Per ogni dipendente ci sarà un identificativo reale: il codice fiscale.  
Al fine di collaborare alla realizzazione del progetto ogni gruppo potrà partecipare a dei meeting, telematici o fisici.  
Il project manager deve poter accedere ai registri delle presenze dei propri colleghi, alle loro skill e al loro salario al fine dell’organizzazione dei gruppi e della valutazione aziendale di ogni singolo membro.

**Capitolo 2**

**Progettazione concettuale**

1. **Analisi di un class diagram**

Un class diagram è un diagramma che descrive la struttura di un sistema. Per apprendere appieno cosa è un class diagram bisogna capire da cosa è formato.  
 Le **classi** rappresentano delle entità. Il nome della classe indica la categoria di entità descritta. Ogni classe è corredata da un insieme di **attributi**, che descrivono in maniera univoca le caratteristiche o lo stato degli oggetti della classe, e **metodi** che descrivono il comportamento di un’entità.  
 Gli attributi sono definiti da un nome, un tipo e una molteplicità.  
Il simbolo grafico che rappresenta le classi è un rettangolo suddiviso in tre scomparti, rispettivamente dedicati al nome della classe, agli attributi e metodi.

Due classi possono essere legate da **relazioni**, esistono diversi tipi di relazione:

* **associazioni:** indicano che due classi (o più nel caso di associazione ternaria) condividono una relazione statica. Le associazioni sono identificate nel class diagram da una o più rette consecutive che collegano due o più classi.
* **specializzazioni** indicano che una delle due classi è una generalizzazione dell’altra (superclasse), ovvero che eredita le caratteristiche (attributi e metodi) della superclasse e le fa proprie.   
  Le specializzazioni sono identificate da una freccia () con la punta verso la classe padre.  
  Esistono diversi tipi di specializzazione:

1. *Totale*: la classe padre è obbligatoriamente una delle classi figlie.
2. *Parziale*: la classe padre può non essere nessuna delle classi figlie.
3. *Disgiunt*a: la classe padre non può essere più classi figlie insieme.
4. *Overlap*: la classe padre può essere più classi figlie insieme.

**2.2 Class Diagram**

Al fine di semplificare la lettura si è scelto di:

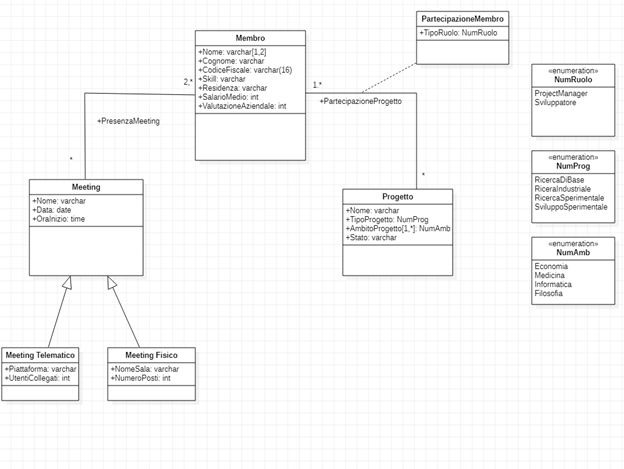
● Non specificare la cardinalità [1] degli attributi in quanto implicito.

● Esplicitare le enumerazioni al lato destro.

● Non impostare la lunghezza degli attributi di tipo VARCHAR a meno che non sia di fondamentale importanza, come nel caso del codice fiscale.

● Le cardinalità delle associazioni indicate con \* specificano che quella classe può partecipare da un minimo di zero a un massimo di molti (0,\*).

● Dare un solo nome alle associazioni invece di due (uno per direzione).



**2.3 Alcune precisazioni sul class diagram**

Dal precedente class diagram si può osservare che sono state create tre classi principali per le entità **membro**, **progetto**, **meeting** e due classi di specializzazione di meeting: **meeting telematico** e **meeting fisico**.

Tra meeting e membro è stata inserita l’associazione **PresenzaMeeting**, questa associazione è chiamata così perché esplicita:

* quanti membri partecipano ai meeting con cardinalità (2,\*) poiché un meeting con meno di due membri non ha senso di esistere.
* a quanti meeting un membro ha preso parte con cardinalità (0,\*) perché se un membro lavora a un progetto da solo (cosa che è possibile fare secondo l’associazione PartecipazioneProgetto) non ha bisogno di partecipare a meeting.

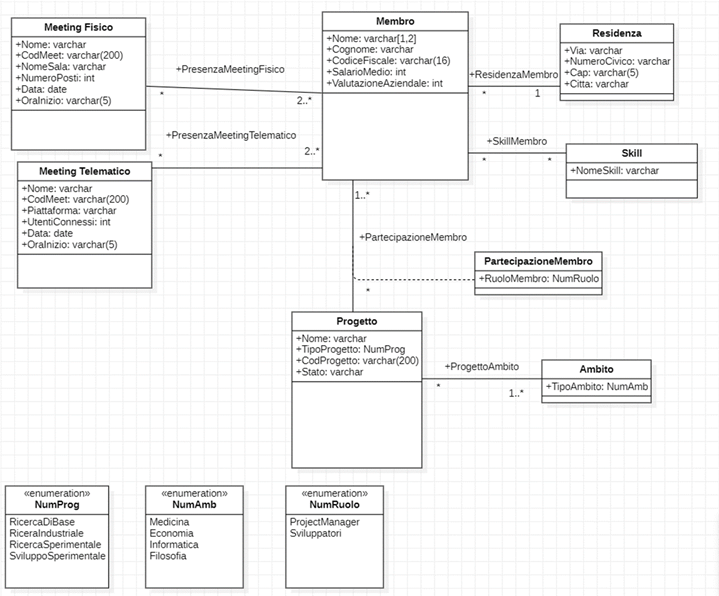
L’associazione **PartecipazioneProgetto** ha il compito di collegare la classe membro e la classe progetto con cardinalità:

* (1,\*) perché un progetto può essere formato da un solo membro.
* (\*) perché un membro può partecipare a nessuno (essere in vacanza o periodo di stop) o più progetti.

**2.4 Ristrutturazione del class diagram**

Al fine di rendere il class diagram idoneo alla traduzione in schemi relazionali e di migliorare l’efficienza dell’implementazione si procede alla ristrutturazione dello stesso. È necessario analizzare le chiavi primarie e fare in modo che siano univoche e più sintetiche possibili. Anche gli attributi derivati, se possibile, devono essere omessi.

Al termine del procedimento di ristrutturazione il class diagram non conterrà attributi strutturati, attributi multipli, gerarchie di specializzazione e associazioni.



1. **Chiavi Primarie**  
    Nel class diagram ristrutturato si è deciso, al fine di agevolare l’identificazione, di aggiungere un attributo destinato a essere la **chiave primaria** della classe qualora la chiave primaria fosse composta.  
   Ad esempio per le classi “progetto” e “meeting” sono state aggiunte le chiavi primarie “CodMeet” e “CodProgetto”, invece per la classe residenza c’è Pod che corrisponde ad un codice identificativo delle abitazioni.
2. **Attributi Multipli**  
    È necessario gestire gli **attributi multipli** in quando questi prevedono la possibilità di contenere più valori. A causa dell’impossibilità di stimare il numero di diversi attributi associati creeremo una nuova classe contenente l’attributo.  
   Per esempio la classe “Membro” contiene l’attributo “Skill”, quest’ultimo potrebbe essere un attributo multiplo in quanto a un membro potrebbero essere associate più skills. Per evitare di avere un attributo di cui non si sa con precisione a che valore riferirsi, si crea una classe “Skill” apposita.  
   Un altro attributo multiplo proviene dalla classe “Progetto”: ambito, infatti un progetto può trattare più ambiti. Anche questa volta creiamo una classe per “ambito”. Facendo ciò però causeremo un’associazione molti a molti, poiché un progetto può includere più ambiti e un ambito è trattato da più progetti.
3. **Attributi Strutturati**  
    Gli attributi strutturati, come gli attributi multipli, devono essere rimossi nel class diagram ristrutturato in quanto questi creano dei problemi quando si desidera conoscere solo una parte dell’attributo.  
   Nel nostro Class Diagram abbiamo l’attributo strutturato “Residenza” ,nella classe “membro”,che è composto dal via, numero civico, cap e dalla città di residenza del membro.

Ci sono due modi per risolvere questa ambiguità: il primo modo è di creare tanti attributi per tutti i campi dell’attributo strutturato, il secondo modo è creare una nuova classe “Residenza” avente gli attributi che corrispondono ai campi del vecchio attributo strutturato.

In questo caso si è scelto il secondo metodo, in quanto è possibile sia accedere alla classe Residenza avendo tutti i campi inclusi, sia accendere a un unico attributo della classe.

1. **Attributi Derivati**

Gli attributi derivati sono attributi derivabili o calcolabili da altri. Un esempio potrebbe essere la data di nascita e l’età, è chiaro che l’età è un attributo derivabile dalla data di nascita. Possiamo rimuovere un attributo derivato nel class diagram, ma possiamo anche mantenere tale attributo per migliorare la comprensibilità.   
Nel class diagram mostrato non ci sono attributi derivati.

1. **Gerarchie di Specializzazione**

Nel primo class diagram si hanno due tipi di specializzazioni. Esistono più modi per eliminare le specializzazioni, si è scelto di accorpare la classe padre nelle classi figlie per la generalizzazione “Meeting”, infatti non troviamo più la classe “Meeting” ma i suoi attributi sono stati trasferiti nelle due classi figlie.

**Capitolo 3**

**Dizionario dei dati**

1. **Dizionario delle Classi**

|  |  |  |
| --- | --- | --- |
| **Classe** | **Descrizione** | **Attributi** |
| Membro | La classe membro corrisponde agli impiegati di un’azienda. | -**Nome** (varchar[1,2]): stringa contenente il nome dell’impiegato, è possibile inserire anche il secondo nome se l’impiegato lo possiede.  -**Cognome** (varchar): stringa del cognome dell’impiegato.  -**CodiceFiscale**(varchar(16)): un codice alfanumerico di 16 cifre che riconosce gli italiani. È quindi il codice identificativo dei membri (primary key).  **-ValutazioneAziendale** (integer): la valutazione affidata al singolo membro  -**SalarioMedio** (integer): è un intero contenente lo stipendio medio (la somma dei salari annuali/12) di un dipendente. |
| Progetto | Progetto è la classe che indica il lavoro svolto o da svolgere dagli impiegati. | **-Nome** (varchar): la stringa contiene il nome del progetto.  -**TipoProgetto** (NumProg) indica il tipo di progetto che devono svolgere o hanno svolto gli impiegati. I tipi possibili sono elencati nell’enumerazione e sono: Ricerca di Base, Ricerca Industriale, Ricerca Sperimentale e Sviluppo Sperimentale.  -**CodProgetto** (varchar(200)): codice identificativo dei progetti(primary key). La lunghezza massima del codice è 200.  -**Stato**(varchar): indica se il progetto è concluso, in svolgimento o in pausa.  -**Ambito** (NumAmb): indica l’ambito del progetto. Gli ambiti possibili sono elencati nell’enumerazione e sono: medicina, economia, filosofia ed informatica. |
| PartecipazioneMembro | È una classe d’associazione che ci aiuta a capire il ruolo di un membro all’interno di un progetto. | -**RuoloMembro**(NumRuolo):indica il ruolo che un membro occupa in un progetto. Tutti i tipi di membri possibili sono elencati nell’enumerazione e sono: Project Manager e Sviluppatore.  Il project manager è un impiegato che ha il ruolo di scegliere i membri di un progetto. Lo sviluppatore è colui che deve portare al termine dei compiti al fine di concludere il progetto a cui è stato assegnato. |
| Meeting Fisico | La classe meeting fisico indica gli incontri che gli impiegati di un’azienda effettuano in una sala riunioni. | -**Nome** (varchar): la stringa contiene il nome del meeting  -**Data** (date): indica la data in cui si tiene il meeting, il tipo data è nel formato AAAA-MM-GG .  -**NomeSala** (varchar): indica il nome della sala dove avviene il meeting.  -**NumeroPosti** (integer): contiene il numero dei posti disponibili per il meeting nella sala scelta.  -**CodMeet** (varchar(200)): è un codice identificativo dato per identificare univocamente la riunione (primary key).  -**OraInizio** (varchar(5)): indica l’ora stabilita per l’inizio dell’incontro, contiene 5 caratteri poiché si è pensato di inserire le ore(due caratteri) e i minuti (due caratteri) separati da due punti “:” (un carattere). |
| Meeting Telematico | La classe meeting telematico, indica gli incontri che gli impiegati di un’azienda effettuano su una piattaforma telematica. | **-Nome** (varchar): la stringa contiene il nome del meeting  -**Data** (date): indica la data in cui si tiene il meeting, il tipo data è nel formato AAAA-MM-GG .  -**OraInizio** (varchar(5)): indica l’ora stabilita per l’inizio dell’incontro, contiene 5 caratteri poiché si è pensato di inserire le ore(due caratteri) e i minuti (due caratteri) separati da due punti “:” (un carattere).  **-CodMeet** (varchar(200)): è un codice identificativo per identificare univocamente la riunione.  -**UtentiConnessi** (integer): indica il numero di utenti connessi al meeting telematico.  -**Piattaforma** (varchar): è il nome della piattaforma usata dagli impiegati per collegarsi al meeting. |
| Residenza | Questa classe contiene delle informazioni riguardanti la residenza di un membro. | **-Via**(varchar): indica la via di residenza.  **-NumeroCivico**(varchar): indica il numero civico dell’abitazione di residenza.  **-Cap** (varchar(5)): indica il cap, ovvero un codice di avviamentopostale di 5 cifre utilizzato per distinguere i comuni italiani.  **-Città** (varchar): indica la città in cui è situata l’abitazione di residenza del membro. **-POD** (varchar(9)):codice di 9 cifre identificativo delle abitazioni |

1. **Dizionario delle associazioni**

|  |  |  |
| --- | --- | --- |
| **Associazioni** | **Descrizione** | **Classi coinvolte** |
| PartecipazioneMembro | Esprime la partecipazione di un impiegato a un progetto. | **-Membro** [1,\*]: indica che a un progetto possono partecipare un membro o più. **-Progetto** [\*]: indica che un membro può partecipare a più progetti. |
| PresenzaMeetingFisico | L’associazione esprime la presenza dei membri ai meeting che avvengono fisicamente. | **-Membro** [2,\*]: a un meeting possono partecipare da 2 a più membri. **-Meeting** [\*]: un membro può aver partecipato a più meeting. |
| PresenzaMeetingTelematico | L’associazione esprime la presenza dei membri ai meeting che avvengono telematicamente. | **-Membro** [2,\*]: a un meeting possono partecipare da 2 a più membri.  **-Meeting** [\*]: un membro può aver partecipato a più meeting. |
| ProgettoAmbito | Esprime l’ambito del progetto. | **-Progetto** [1,\*]: a un progetto possono essere associati da 1 a più ambiti.  **-Ambito** [\*]: un ambito può essere associato a più progetti. |
| SkillMembro | Questa associazione esplicita le skill che un membro possiede. | **-Skill**[\*]: una skill può appartenere a nessuno o a più membri.  **-Membro**[\*]: a un membro può avere nessuna o più skills. |
| ResidenzaMembro | Questa associazione serve a fare chiarezza sul luogo di residenza di un membro. | **-Residenza**[\*]: una residenza può appartenere a nessuno o a più membri.  **-Membro**[1]:un membro ha un’unica residenza. |

1. **Dizionario dei vincoli**

|  |  |  |
| --- | --- | --- |
| **Vincoli** | **Attributo** | **Descrizione** |
| Legit Nome Membro | Membro(Nome) | Il nome del membro può contenere solo caratteri a A-Z o a-z. |
| Unicità CodiceFiscale | Membro(Codice Fiscale) | Ogni membro deve avere al più un codice fiscale. Non possono esistere due membri con lo stesso codice fiscale. |
| Legit CodiceFiscale | Membro(Codice Fiscale) | L’attributo codice fiscale è composto da 16 caratteri tra cui i primi sei caratteri (A-Z) riferiti al cognome e nome.  I successivi due (0-9) si riferiscono all’anno di nascita, seguiti da uno (A-Z) che si riferisce al mese di nascita e due (0-9) al giorno.  Successivamente ci sono 4 caratteri per il comune, e uno per il codice controllo (A-Z). |
| Legit Valutazione | Membro (ValutazioneAziendale) | La valutazione deve essere compresa tra un minimo di 0 e un massimo di 10 |
| Legit Via | Residenza(Via) | La via deve essere una via esistente nel comune di residenza indicato nel codice fiscale. |
| Legit Cap | Residenza(Via) | Il cap deve corrispondere al cap del comune indicato nel codice fiscale o/e a un comune della città indicata. |
| Legit Città | Residenza(Città) | La città deve corrispondere alla città di appartenenza del comune indicato sia nel codice fiscale sia nel cap. |
| Legit Nome Progetto | Progetto(Nome) | Il nome del progetto può contenere solo caratteri a A-Z o a-z. |
| Unicità CodProgetto | Progetto(CodProgetto) | Ogni progetto deve avere al più un codice progetto, non esistono due progetti con lo stesso codice. |
| Legit NomeSala | Meeting Fisico(NomeSala) | Il nome deve essere un nome di una sala esistente dell’azienda. |
| Legit Piattaforma | Meeting Telematico | Il nome deve essere il nome di una piattaforma che permette meeting. |
| Legit CodMeet Telematico | Meeting Telematico (CodMeet) | Ogni meeting deve avere al più un codice meeting, non esistono due meeting con lo stesso codice. |
| Legit CodMeet Fisico | Meeting Fisico(CodMeet) | Ogni meeting deve avere al più un codice meeting, non esistono due progetti con lo stesso meeting. |

Capitolo 4|Progettazione Logica

In questo capitolo verrà tradotto lo schema concettuale in uno schema logico, dipendente dal tipo di struttura dei dati prescelto cioè quello relazionale puro.   
Si precisa che negli schemi relazionali che seguiranno le chiavi primarie sono indicate con una singola sottolineatura mentre le chiavi esterne con una doppia sottolineatura.

1. Schemi Logici  
   **Membro**(CodFiscale, Nome, Cognome, ValutazioneAziendale, SalarioMedio, Pod)  
   Chiave esterna: Pod→Residenza.Pod

**Progetto**(CodProgetto, Nome, TipoProgetto, Stato)

**MeetingFisico**(Nome, CodMeet, NomeSala, NumeroPosti, Data, OraInizio)

**MeetingTelematico**(Nome, CodMeet, Piattaforma, UtentiConnessi, Data, OraInizio)

**Ambito**(TipoAmbito)

**AmbitoProgetto**(TipoAmbito, CodProgetto)  
 Chiave esterna: TipoAmbito→Ambito.TipoAmbito  
 Chiave esterna: CodProgetto→Progetto.CodProgetto

**ArchivioPartecipazioni**(CodiceFiscale, CodProgetto)  
 Chiave esterna: CodiceFiscale→Membro.CodiceFiscale  
 Chiave esterna: CodProgetto→Progetto.CodProgetto

**ArchivioPresenzeTelematiche**(CodiceFiscale, CodMeet)  
 Chiave esterna: CodiceFiscale→Membro.CodiceFiscale  
 Chiave esterna: CodMeet→MeetingTelematico.CodMeet

**ArchivioPresenzeFisiche**(CodiceFiscale, CodMeet)  
 Chiave esterna: CodiceFiscale→Membro.CodiceFiscale  
 Chiave esterna: CodMeet→MeetingFisico.CodMeet

**Skill**(NomeSkill,CodFiscale)  
 Chiave esterna: NomeSkill→Skill.NomeSkill  
 Chiave esterna: CodiceFiscale→Membro.CodiceFiscale

**Residenza**(POD, Via, NumeroCivico, Cap, Città)

Capitolo 5|Progettazione Fisica (SQL)

L'ultima fase della progettazione di una basi di dati è la **progettazione fisica**. Prima di iniziare la progettazione fisica occorre scegliere un DBMS (DataBase Management System, Sistema di Gestione di DataBase) che implementi il modello dei dati dello schema logico.

Il DBMS è uno software che si frappone tra l’utente e i dati, i dbms non accedono ai dati così come sono memorizzati effettivamente, ma ne vedono solamente una rappresentazione logica.  
Il dbms usato in questo caso è **PostgreSQL.**  
**SQL** (Structured Query Language) è il linguaggio di interrogazione più diffuso tra quelli usati per l’interazione con i principali DBMS.

1. Definizione tabelle

Creazione tabella membro

1. CREATE TABLE Membro(
2. Nome VARCHAR NOT NULL,
3. Cognome VARCHAR NOT NULL,
4. CodiceFiscale VARCHAR(16) NOT NULL CHECK ( CodiceFiscale LIKE '[A-Z][A-Z][A-Z][A-Z][A-Z][A-Z][0-9][0-9][A-Z][0-9][0-9][A-Z][0-9][0-9][0-9][A-Z]'),
5. SalarioMedio INT NOT NULL,
6. ValutazioneAziendale INTEGER );