

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE
DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI

DOCUMENTAZIONE DEL PROGETTO DI
BASI DI DATI:
PROGETTAZIONE E IMPLEMENTAZIONE
GRUPPO: OOB7

Docente

Prof. Mara Sangiovanni

Candidati

Virginia Antonia Esposito
(N86/004987)

Giuseppe Paolo Esposito
(N86/005174)

Anno Accademico 2024/2025

Contents

1	Introduzione e Analisi del Problema	2
2	Progettazione schema della base di dati	3
2.1	Diagramma delle classi UML	3
2.2	Diagramma ER (Entità Relazione)	4
3	Ristrutturazione dello schema secondo il modello relazionale	6
3.1	Diagramma delle classi UML ristrutturato	6
3.2	Motivazione delle Scelte di Ristrutturazione	7
3.3	Dizionario delle Classi, Associazioni e Vincoli	8
4	Schema Logico	13
4.1	Vincoli Referenziali	14
5	Schema fisico	15
5.1	Struttura SQL delle tabelle	15
5.2	Trigger e procedure SQL	20

1 Introduzione e Analisi del Problema

UninaFoodLab nasce come sistema per la gestione di corsi di cucina tematici.

Attraverso questa piattaforma ci si potrà **registrare** come partecipante o chef, i quali potranno rispettivamente partecipare ai corsi di cucina proposti oppure aggiungerne di nuovi.

Gli chef per registrarsi avranno bisogno di inserire un curriculum, così da poter dimostrare di essere chef con esperienza e garantire corsi ben strutturati ai partecipanti.

Gli chef avranno la possibilità di gestire i propri corsi come desiderano, scegliendone la modalità di erogazione, tra online e pratico, il numero e la frequenza delle sessioni, il costo, la data e il luogo.

Nel caso di un corso che presenta anche **sessioni pratiche**, potranno inoltre decidere il numero di partecipanti limite e, per ciascuna delle sessioni pratiche, scegliere tra le proprie ricette quali preparare.

Le ricette possono essere create consultando l'elenco di ingredienti già disponibili globalmente a tutti gli chef oppure, nell'eventualità questi non siano presenti, il sistema permette la creazione di nuovi ingredienti che poi verranno aggiunti.

Per la **creazione** dei corsi potranno essere scelti fino a 5 argomenti, anche questi presenti globalmente per tutti gli chef, rappresentato sotto forma di keyword in modo tale da descriverli al meglio possibile.

I partecipanti potranno, se sono iscritti ad un corso che presenta sessioni pratiche e sono sicuri di partecipare, mandare un'adesione, la quale sarà utile allo chef per calcolare esattamente la quantità di ingredienti necessari.

Il sistema garantisce proprio a questo scopo una gestione coerente della adesioni in modo tale da evitare sprechi per adesioni revocate con poco preavviso.

Anche i partecipanti vengono **tutelati** garantendo il regolare svolgimento dei corsi ai quali sono iscritti, a meno di esigenze particolari dello chef.

2 Progettazione schema della base di dati

2.1 Diagramma delle classi UML

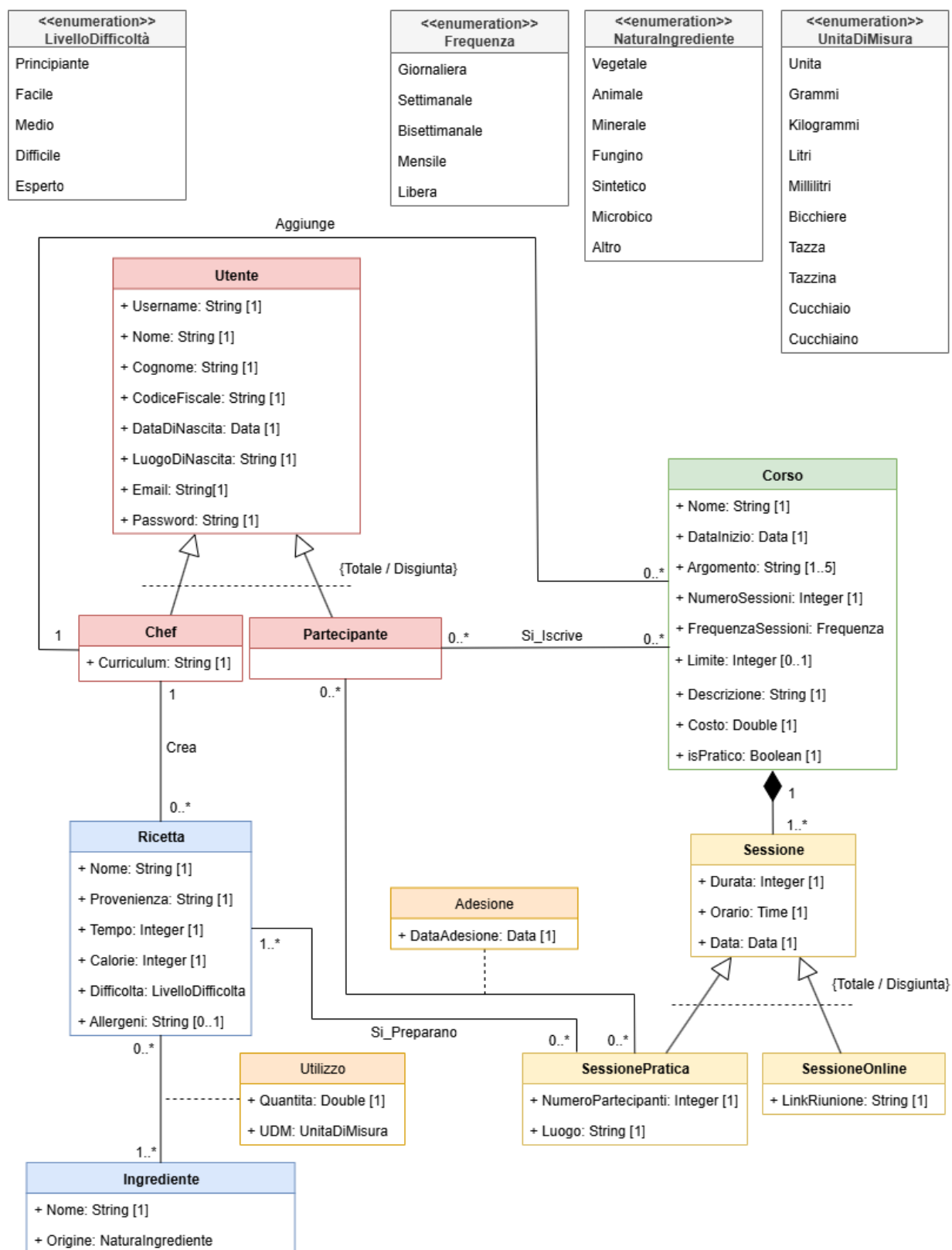


Figure 1: Diagramma delle Classi UML

2.2 Diagramma ER (Entità Relazione)

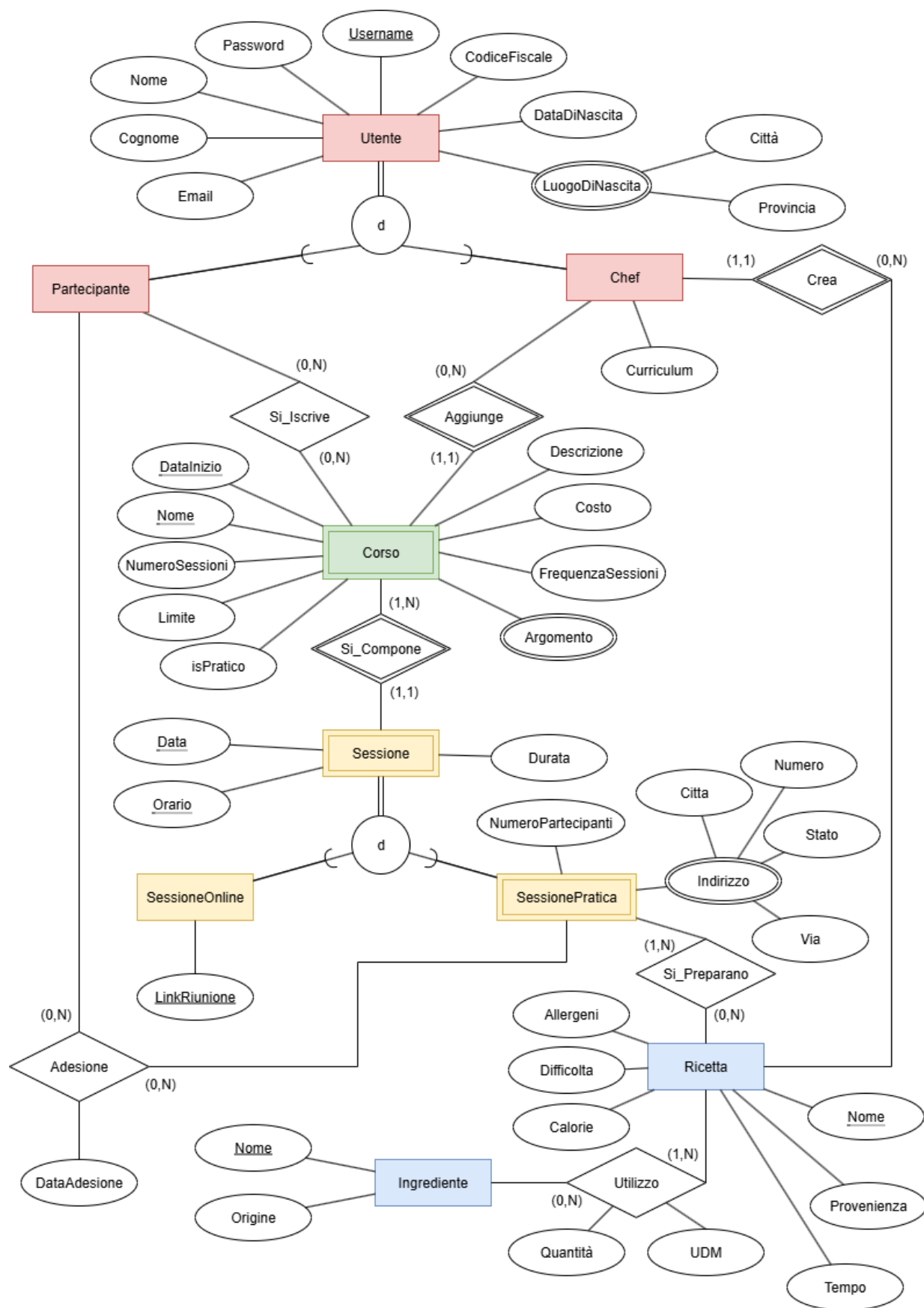


Figure 2: Diagramma Entità-Relazione

3 Ristrutturazione dello schema secondo il modello relazionale

3.1 Diagramma delle classi UML ristrutturato

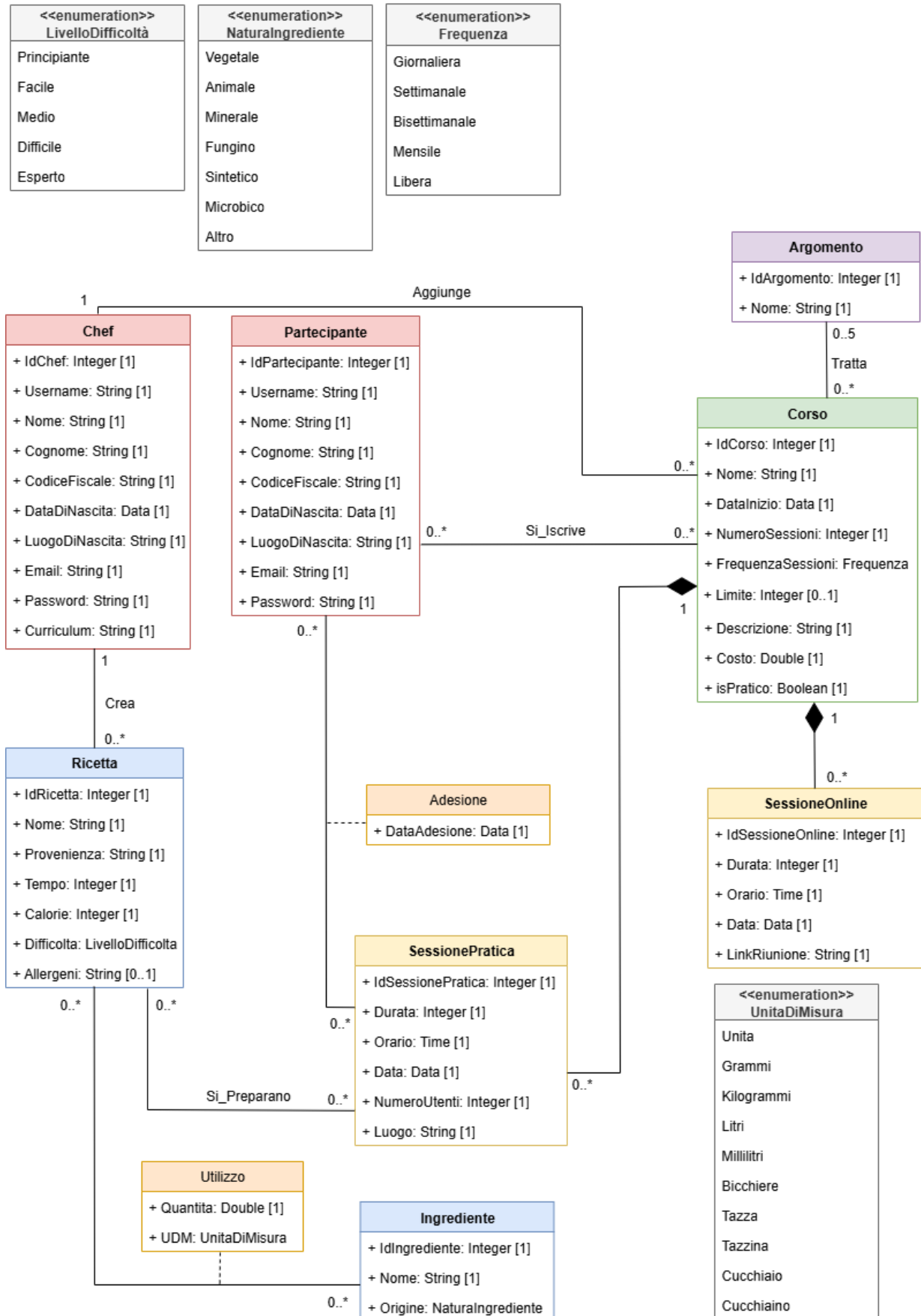


Figure 3: Diagramma UML Ristrutturato

3.2 Motivazione delle Scelte di Ristrutturazione

- **Rimozione degli attributi strutturati:**

- **Utente:** Luogo di nascita che era diviso in città e provincia viene trattato come stringa unica
- **SessionePratica:** Indirizzo che era diviso in via, numero, città, stato viene trattato come stringa unica

- **Rimozione degli attributi multivalore:**

- **Corso:** Argomento viene trattato come classe individuale perchè, dal momento che potremmo avere fino a 5 argomenti, esso costituirebbe una stringa molto lunga da parsare e, siccome gli argomenti che possono essere scelti per un corso sono comuni a più corsi, evitiamo ridondanza creando una classe a parte.

- **Rimozione delle generalizzazioni:**

- **Utente** in Partecipazione e Chef: essendo una generalizzazione totale e disgiunta decido di accorpare la classe generale nelle sue sottoclassi, aggiungendo gli attributi di Utente agli attributi delle rispettive sottoclassi.
- **Sessione** in SessionePratica e SessioneOnline: essendo una generalizzazione totale e disgiunta decido di accorpare la classe generale nelle sue sottoclassi, aggiungendo gli attributi di Sessione agli attributi delle rispettive sottoclassi.

- **Scelta degli identificatori primari:**

- **Partecipante:** *IdPartecipante*, viene aggiunta come chiave surrogata poiché nonostante username, email e codice fiscale siano ciascuno attributi identificanti, sono tutte stringhe che potrebbero essere abbastanza lunghe e risulterebbero difficili da cercare e indicizzare.
- **Chef:** *IdChef*, viene aggiunta come chiave surrogata poiché nonostante username, email e codice fiscale siano ciascuno attributi identificanti, sono tutte stringhe che potrebbero essere abbastanza lunghe e risulterebbero difficili da cercare e indicizzare.
- **Corso:** *IdCorso*, viene aggiunta come chiave surrogata poiché nonostante nome e dataInizio in relazione allo chef sono attributi identificanti, il nome del corso potrebbe essere una stringa abbastanza lunga e risulterebbe difficile da cercare e indicizzare.
- **SessionePratica:** *IdSessionePratica*, viene aggiunta come chiave surrogata, poiché nonostante data e orario in relazione al corso sono attributi identificanti, tener conto di più attributi risulterebbe costoso per ricerca e indicizzazione
- **SessioneOnline:** *IdSessioneOnline*, viene aggiunta come chiave surrogata, poiché nonostante data e orario in relazione al corso e il link della riunione siano attributi identificanti, tener conto di più attributi risulterebbe costoso per ricerca e indicizzazione e il link potrebbe potenzialmente essere una stringa molto lunga.
- **Argomento:** *IdArgomento*, viene aggiunta come chiave surrogata, poiché nonostante il nome sia un attributo identificante, potrebbe essere una stringa potenzialmente molto lunga e risulterebbe difficile da cercare e indicizzare.

Ricetta: *IdRicetta*, viene aggiunta come chiave surrogata, poiché nonostante il nome in relazione allo chef sia un attributo identificante, il nome potrebbe essere una stringa potenzial-

mente molto lunga e risulterebbe difficile da cercare e indicizzare.

- **Ingrediente:** *IdIngrediente*, viene aggiunta come chiave surrogata, poiché nonostante il nome sia un attributo identificante, il nome potrebbe essere una stringa potenzialmente molto lunga e risulterebbe difficile da cercare e indicizzare.

- **Analisi delle ridondanze:**

Sono presenti due cicli nel modello concettuale:

1) **Chef -> Corso -> SessionePratica -> Ricetta -> Chef**: In questo ciclo è presente una ridondanza parziale perchè, tramite la relazione *Si_Preparano*, avremmo già le informazioni sulle ricette dello chef che sono presenti nella sessione pratica. Non avremmo però informazione sulle ricette che sono state create dallo chef e che non sono state assegnate a una sessione pratica. Quindi, nonostante questa ridondanza, la relazione *Crea* è necessaria al fine di tracciare tutte le ricette dello chef.

2) **Partecipante -> Corso -> SessionePratica -> Partecipante**: Per questo ciclo non sono presenti ridondanze perchè la relazione *Si_Iscrive* da informazioni solo sulla iscrizione al corso (che è composizione di sessioni) ma non sulla eventuale adesione del partecipante alle sessioni pratiche. È quindi una relazione che esprime una informazione diversa e che è necessaria al fine di tracciare le adesioni alle sessioni pratiche.

- **Attributi derivabili:**

Per quanto riguarda l'attributo *NumeroSessioni* in *Corso*, esso potrebbe essere derivabile dal conteggio delle occorrenze nella *SessionePratica* e *SessioneOnline* con lo stesso *IdCorso*. Esso però rappresenta un'informazione concettualmente rilevante per l'entità *Corso*, che viene consultata frequentemente dagli utenti e utilizzata in varie operazioni, inoltre per calcolare questo valore sarebbe necessario contare le istanze su due entità distinte e per corsi con molte sessioni questo diventa inefficiente. Lo teniamo quindi memorizzato e aggiornato tramite operazioni automatiche. Lo stesso discorso vale per l'attributo *NumeroPartecipanti* in *SessionePratica* che potrebbe essere derivabile dal conteggio delle istanze nella relazione *Adesione*.

- **Partizionamento/Accorpamento di entità ed associazioni:**

Non sono presenti accorpamenti o partizionamenti di entità o associazioni.

3.3 Dizionario delle Classi, Associazioni e Vincoli

Dizionario delle Classi

Considereremo come default gli attributi come totali; nella tabella verranno indicati i casi in cui sono parziali.

Nome	Descrizione	Attributi
Partecipante	Rappresenta un utente che partecipa ai corsi tenuti dagli chef	IdPartecipante (Integer): Chiave virtuale, identifica univocamente ogni partecipante Username (String): Rappresenta l'username scelto dal partecipante alla registrazione Nome (String): Nome del partecipante Cognome (String): Cognome del partecipante CodiceFiscale (String): Codice Fiscale del partecipante DataDiNascita (Data): Data di nascita del partecipante LuogoDiNascita (String): Luogo di nascita del partecipante Email (String): Rappresenta l'email scelta dal partecipante alla registrazione Password (String): Password
Chef	Rappresenta un utente che crea ricette con ingredienti, tiene i corsi e organizza le sessioni	IdChef (Integer): Chiave virtuale, identifica univocamente ogni chef Username (String): Rappresenta l'username scelto dallo chef alla registrazione Nome (String): Nome dello chef Cognome (String): Cognome dello chef CodiceFiscale (String): Codice Fiscale dello chef DataDiNascita (Data): Data di nascita dello chef LuogoDiNascita (String): Luogo di nascita dello chef Email (String): Rappresenta l'email scelta dallo chef alla registrazione Password (String): Password scelta dallo chef alla registrazione Curriculum (String): Path del file del curriculum dello chef
Ricetta	Rappresenta una ricetta creata dallo chef	IdRicetta (Integer): Chiave virtuale, identifica univocamente ogni ricetta Nome (String): Nome della ricetta Provenienza (String): Luogo originario della ricetta Tempo (Integer): Minuti necessari alla preparazione Calorie (Integer): Kilocalorie totali della ricetta Difficoltà (LivelloDifficoltà): Principiante, Facile, Medio, Difficile, Esperto Allergeni (String – Parziale): Elenca eventuali allergeni
Utilizzo	Rappresenta una quantità di ingrediente utilizzata nella ricetta di uno chef	Quantita (Double): Numero che, con UDM, indica l'utilizzo totale UDM (UnitaDiMisura): Unità di misura (Unità, Grammi, ..., Cucchiaino)
Ingrediente	Rappresenta un ingrediente utilizzato nella ricetta di uno chef	IdIngrediente (Integer): Chiave virtuale, identifica univocamente ogni ingrediente Nome (String): Nome dell'ingrediente Origine (NaturaIngrediente): Vegetale, Animale, ..., Altro
Corso	Rappresenta un corso creato da uno chef	IdCorso (Integer): Chiave virtuale Nome (String): Nome del corso DataInizio (Data): Data di inizio del corso NumeroSessioni (Integer): Numero delle sessioni FrequenzaSessioni (Frequenza): Giornaliera, Settimanale, ..., Libera Limite (Integer – Parziale): Limite per corsi pratici Descrizione (String): Descrizione del corso Costo (Double): Costo del corso isPratico (Boolean): True se ha sessioni pratiche
Argomento	Rappresenta una tematica di un corso	IdArgomento (Integer): Chiave virtuale Nome (String): Nome dell'argomento
SessioneOnline	Rappresenta una sessione online di un corso	IdSessioneOnline (Integer): Chiave virtuale Durata (Integer): Durata della sessione online Orario (Time): Orario previsto Data (Data): Data della sessione online LinkRiunione (String): Link della riunione

Continua nella pagina successiva

Nome	Descrizione	Attributi
SessionePratica	Rappresenta una sessione pratica di un corso	IdSessionePratica (Integer): Chiave virtuale Durata (Integer): Durata della sessione pratica Orario (Time): Orario previsto Data (Data): Data della sessione pratica NumeroUtenti (Integer): Numero di adesioni Luogo (String): Luogo della sessione pratica
Adesione	Rappresenta un'adesione di un partecipante a una sessione pratica	DataAdesione (Data): Data in cui è stata effettuata l'adesione

Dizionario delle Associazioni

Nome	Descrizione	Classi e Ruoli
Aggiunge	Esprime l'aggiunta di un corso da parte dello chef all'applicazione.	Chef [0..*] ruolo (aggiunge): indica che lo chef può aggiungere più corsi. Corso [1] ruolo (è aggiunto): indica che un corso è aggiunto da un solo chef.
Crea	Esprime la creazione di una ricetta da parte dello chef.	Chef [0..*] ruolo (crea): indica che lo chef può creare più ricette. Ricetta [1] ruolo (è creata): indica che una ricetta è creata da un solo chef.
Utilizzo	Esprime la quantità utilizzata da una ricetta di uno specifico ingrediente.	Ricetta [0..*] ruolo (utilizza): indica che una ricetta può usare più ingredienti. Ingrediente [0..*] ruolo (si utilizza): indica che un ingrediente può essere usato in più ricette. Classe di associazione Utilizzi con attributi: Quantità, UDM.
Si_Preparano	Esprime la possibilità di preparare una ricetta nella sessione pratica.	Ricetta [0..*] ruolo (è preparata): indica che una ricetta può essere preparata in più sessioni pratiche. SessionePratica [0..*] ruolo (si preparano): indica che in una sessione pratica possono essere preparate più ricette.
Tratta	Esprime una relazione tra corso e argomenti.	Argomento [0..*] ruolo (è trattata): indica gli argomenti di un corso. Corso [0..*] ruolo (tratta): indica i corsi relativi a un argomento.
Adesione	Esprime una relazione tra partecipanti e sessioni pratiche.	Partecipante [0..*] ruolo (aderisce): indica che un partecipante aderisce a più sessioni pratiche. SessionePratica [0..*] ruolo (è aderita): indica che una sessione pratica riceve adesioni da più partecipanti. Classe di associazione Adesioni con attributo: DataAdesione.
Si_Iscrive	Esprime una relazione tra partecipanti e corsi.	Partecipante [0..*] ruolo (si iscrive): indica che un partecipante può iscriversi a più corsi. Corso [0..*] ruolo (ha iscrizioni): indica che un corso può avere iscrizioni da più partecipanti.
Composizione: Corso-SessionePratica	Relazione di composizione tra corso e sessione pratica.	SessionePratica [1] ruolo (fa parte di): ogni sessione pratica fa parte di un solo corso. Corso [0..*] ruolo (contiene): un corso può contenere più sessioni pratiche.
Composizione: Corso-SessioneOnline	Relazione di composizione tra corso e sessione online.	SessioneOnline [1] ruolo (fa parte di): ogni sessione online fa parte di un solo corso. Corso [0..*] ruolo (contiene): un corso può contenere più sessioni online.

Table 2: Dizionario dei Vincoli

Nome	Tipo	Descrizione
unique_username_partecipante	Intrarelazionale	L'username di un partecipante deve essere unico
unique_codicefiscale_partecipante	Intrarelazionale	Il codice fiscale di un partecipante deve essere unico
unique_email_partecipante	Intrarelazionale	L'email di un partecipante deve essere unica
check_empty_part_nome	Dominio	Il nome del partecipante non può essere vuoto
check_empty_part_cognome	Dominio	Il cognome del partecipante non può essere vuoto
check_correct_part_cf	Dominio	Il codice fiscale del partecipante deve avere lunghezza 16
check_empty_part_luogonascita	Dominio	Il luogo di nascita del partecipante non può essere vuoto
check_empty_part_email	Dominio	L'email del partecipante non può essere vuota
check_empty_part_pass	Dominio	La password del partecipante non può essere vuota
check_part_length	Dominio	La lunghezza dell'username del partecipante deve essere tra 4 e 20 caratteri
check_part_maggiorenne	Dominio	Controlla che il partecipante sia maggiorenne
unique_username_chef	Intrarelazionale	L'username di uno chef deve essere unico tra tutti gli chef
unique_codicefiscale_chef	Intrarelazionale	Il codice fiscale deve essere unico tra gli chef
unique_email_chef	Intrarelazionale	L'email deve essere unica tra gli chef
check_empty_chef_nome	Dominio	Il nome deve aver una lunghezza maggiore di 0
check_empty_chef_cognome	Dominio	Il cognome deve avere una lunghezza maggiore di 0
check_correct_chef_cf	Dominio	Il codice fiscale deve avere una lunghezza pari a 16
check_empty_chef_luogonascita	Dominio	Il luogo di nascita deve avere una lunghezza maggiore di 0
check_empty_chef_email	Dominio	L'email deve avere una lunghezza maggiore di 0
check_empty_chef_pass	Dominio	La password deve avere lunghezza maggiore di 0
check_empty_chef_curriculum	Dominio	Il path del curriculum deve avere una lunghezza maggiore di 0
check_chef_length	Dominio	La lunghezza dell'username deve essere compresa tra 4 e 20
check_chef_maggiorenne	Dominio	Lo chef deve essere maggiorenne
unique_nome_argomento	Intrarelazionale	Il nome dell'argomento deve essere unico per evitare duplicati
check_empty_corso_nome	Dominio	Il nome del corso non può essere vuoto
check_data_non_passata_corso	Dominio	Il corso non può avere una data di inizio nel passato
check_empty_corso_descr	Dominio	La descrizione del corso non può essere vuoto
check_costo_non_negativo	Dominio	Il costo del corso non può essere negativo
check_limite_pratico	N-Upla	Il corso ha un limite solo se è pratico, altrimenti il campo limite è nullo
check_data_sessioneprat_curr	Dominio	La data della sessione pratica deve essere maggiore della data corrente
check_data_sessioneprat_0	Dominio	La durata di una sessione pratica deve essere maggiore di 0
check_empty_sessioneprat_luogo	Dominio	La lunghezza del luogo deve essere maggiore di 0
check_data_sessioneonl_curr	Dominio	La data della sessione online deve essere maggiore della data corrente
check_data_sessioneonl_0	Dominio	La durata di una sessione online deve essere maggiore di 0
check_empty_sessioneonl_link	Dominio	La lunghezza del link deve essere maggiore di 0
unique_nome_chef	Intrarelazionale	Una ricetta dello stesso chef non può avere lo stesso nome
check_empty_ricetta_nome	Dominio	Il nome della ricetta non può essere vuoto
check_empty_ricetta_provenienza	Dominio	La provenienza della ricetta non può essere vuota
check_tempo_positivo	Dominio	Il tempo della provenienza deve essere positivo
check_calorie_positive	Dominio	Le calorie della ricetta devono essere positive
check_empty_ingr_nome	Dominio	La lunghezza del nome deve essere maggiore di 0
unique_nome_ingredient	Interrelazionale	Il nome dell'ingrediente deve essere unico nella tabella ingredienti
pk_iscrizioni_corso	Intrarelazionale	Lo stesso utente non può partecipare più di una volta allo stesso corso allo stesso momento
pk_argomenti_corso	Intrarelazionale	Non ci possono essere argomenti ripetuti per un corso
pk_adesioni	Intrarelazionale	Lo stesso partecipante non può aderire più di una volta alla stessa sessione pratica allo stesso momento
pk_sessionepratica_ricetta	Intrarelazionale	Non si può ripetere la stessa ricetta nella stessa sessione
pk_ricetta_ingredient	Intrarelazionale	Gli ingredienti non devono essere ripetuti per la stessa ricetta
check_quantita	Dominio	La quantità di un ingrediente deve essere positiva
uniqueUsernamePartChef	Interrelazionale	Impedire che due utenti – uno Chef e uno Partecipante – abbiano lo stesso Username, anche se sono in due tabelle diverse
isPraticoCheck	Interrelazionale	Non posso inserire una sessione pratica se IsPratico = false
unicitàSessioneGiorno	Interrelazionale	Non ci possono essere più sessioni per lo stesso corso nello stesso giorno
dataSessioneDopoDataCorso	Interrelazionale	La data della sessione non può essere messa prima della data inizio corso

Continua nella pagina successiva

Nome	Tipo	Descrizione
dataPrimaSessione	Interrelazionale	Se non ci sono sessioni per quel corso allora la data della prima sessione inserita deve essere inserita il giorno di inizio corso
verificaFrequenzaCorso	Interrelazionale	Quando inserisco o modifico una sessione essa deve rispettare l'intervallo della frequenza del corso a cui appartiene
controllaSovrapposizioniOrariSessioni	Interrelazionale	Quando si inserisce una sessione per un corso, essa non può avvenire nella stessa fascia oraria di un'altra sessione dello chef per quel giorno
checkNumArgomentiCorso	Intrarelazionale	Gli argomenti associati ad un corso non possono essere più di 5
checkLimiteCorso	Interrelazionale	Se il corso è pratico, bisogna controllare che non venga raggiunto il limite di iscrizioni
checkIscrizioneBeforeAdesione	Interrelazionale	Un partecipante non può aderire a una sessione pratica se non iscritto al corso che la organizza
checkAdesioneOnTime	Interrelazionale	La data dell'adesione alla sessione pratica deve essere antecedente (massimo 3 giorni prima) alla data della sessione pratica
checkChefRicettaCorso	Interrelazionale	Controllare che lo chef della ricetta aggiunta alla sessione pratica sia lo stesso dello chef che organizza il corso
immutableCodiceFiscale	Intrarelazionale	Il codice fiscale di un utente non può essere modificato una volta inserito
checkDeletePartecipanteConAdesione	Interrelazionale	Non è possibile eliminare un partecipante se ha aderito a una sessione pratica e siamo a meno di 3 giorni dalla sua data
checkCorsiAttiviBeforeDeleteChef	Interrelazionale	Non è possibile eliminare uno chef se ha dei corsi ancora attivi
updatePermessiCorso	Intrarelazionale	Di un corso è possibile aggiornare solo il Nome, la Descrizione, La Frequenza e il Numero Sessioni
bloccaUpdateOnDataCorso	Interrelazionale	Non è possibile modificare la data di inizio del corso se ci sono sessioni associate
bloccaCancellaCorsoAttivo	Interrelazionale	Non è possibile cancellare un corso attivo ¹
bloccaAggiornamentoCorsoSessione	Intrarelazionale	Non è possibile modificare il corso associato a una sessione già esistente
bloccaCancellaSessioneCorsoAttivo	Interrelazionale	Non puoi cancellare una sessione passata se il suo corso è ancora attivo
bloccaCancAggArgomento	Intrarelazionale	Non si possono eliminare o aggiornare gli argomenti (Tabella globale)
bloccaCancellazioneUltimoArgomento	Intrarelazionale	Non deve essere possibile cancellare l'ultimo argomento di un corso
bloccaUpdateArgomentiCorso	Interrelazionale	E' possibile aggiornare gli argomenti del corso solo se non è ancora iniziato e non ha iscrizioni
bloccaUpdateChefInRicetta	Intrarelazionale	Non è possibile aggiornare lo chef associato a una ricetta
bloccaDeleteRicettaInUtilizzo	Interrelazionale	Non è possibile cancellare una ricetta se è in uso in una sessione pratica di un corso attivo
bloccaCancAggIngredienti	Intrarelazionale	Non è possibile aggiornare o modificare ingredienti (Tabella globale)
bloccaAggiuntaRicetteSessioniFinite	Interrelazionale	Non è possibile aggiungere una ricetta alla sessione pratica se essa è già avvenuta
bloccaDisiscrizioneCorsoSeAderito	Interrelazionale	Non è possibile disiscrivere dal corso se si è aderito a una sessione pratica e mancano meno di 3 giorni dal suo avvenimento
bloccaIscrizioneSeCorsoIniziato	Interrelazionale	Non è possibile iscriversi a un corso se è già iniziato
bloccaUpdatePratico	Interrelazionale	Se ci sono sessioni pratiche, non è possibile cambiare il tipo del corso da pratico a non pratico
bloccaCancellaAdesioneDopo3Giorni	Interrelazionale	Non è possibile eliminare una adesione a una sessione pratica a meno di 3 giorni dalla data in cui essa avviene
bloccaAggiornaDataAdesione	IntraRelazionale	Non è possibile modificare la data di adesione dopo aver aderito
cancellaAdesioniSeDisiscritto	Interrelazionale	Se è possibile disiscrivere dal corso vengono tolte le adesioni effettuate da oggi in poi alle sessioni pratiche di quel corso
onCascadeCorsoFromChef	Interrelazionale	Il corso viene eliminato se viene eliminato lo chef che lo organizza (succede solo se non ha corsi attivi)
onCascadeSessionePraticaFromCorso	Interrelazionale	Una sessione pratica viene eliminata se viene eliminato il corso che la contiene
onCascadeSessioneOnlineFromCorso	Interrelazionale	Una sessione online viene eliminata se viene eliminato il corso che la contiene
onCascadeRicettaFromChef	Interrelazionale	Una ricetta viene eliminata se viene eliminato lo chef che l'ha creata
onCascadeIscrizioniFromPartecipante	Interrelazionale	Se viene cancellato un partecipante, allora vengono eliminate tutte le sue iscrizioni ai corsi
onCascadeIscrizioniFromCorso	Interrelazionale	Se viene cancellato un corso, allora vengono eliminate tutte le iscrizioni a quel corso
onCascadeArgomentiCorsoFromCorso	Interrelazionale	Se viene cancellato un corso, allora vengono anche cancellati i suoi argomenti
onCascadeUtilizziFromRicetta	Interrelazionale	Se viene cancellata una ricetta, allora vengono cancellati anche i suoi utilizzi
onCascadePrepFromRicetta	Interrelazionale	Se viene cancellata una ricetta, allora vengono cancellate anche tutte le sue preparazioni nelle sessioni pratiche

¹D'ora in avanti con corso "attivo" si intende un corso con un numero di iscritti maggiore di 0 non ancora terminato

Continua dalla pagina precedente

Nome	Tipo	Descrizione
onCascadePrepFromSessionePratica	Interrelazionale	Se viene cancellata una sessione pratica, vengono cancellate anche tutte le sue preparazioni
onCascadeAdesFromSessionePratica	Interrelazionale	Se viene cancellata una sessione pratica, vengono cancellate anche le sue adesioni

4 Schema Logico

Entità	Attributi
Partecipante	(<u>IdPartecipante</u> , Username, Nome, Cognome, CodiceFiscale, DataDiNascita, LuogoDiNascita, Email, Password, NumeroCorsi)
Chef	(<u>IdChef</u> , Username, Nome, Cognome, CodiceFiscale, DataDiNascita, LuogoDiNascita, Email, Password, Curriculum)
Corso	(<u>IdCorso</u> , Nome, DataInizio, NumeroSessioni, FrequenzaSessioni, Limite, Descrizione, Costo, isPratico, <u>IdChef</u>)
Argomento	(<u>IdArgomento</u> , Nome)
SessionePratica	(<u>IdSessionePratica</u> , Durata, Orario, Data, NumeroUtenti, Luogo, <u>IdCorso</u>)
SessioneOnline	(<u>IdSessioneOnline</u> , Durata, Orario, Data, LinkRiunione, <u>IdCorso</u>)
Ricetta	(<u>IdRicetta</u> , Nome, Provenienza, Tempo, Calorie, Difficolta, Allergeni, <u>IdChef</u>)
Ingrediente	(<u>IdIngrediente</u> , Nome, Origine)
Iscrizioni	(<u>IdPartecipante</u> , <u>IdCorso</u>)
Argomenti_Corso	(<u>IdCorso</u> , <u>IdArgomento</u>)
Adesioni	(<u>IdPartecipante</u> , <u>IdSessionePratica</u> , DataAdesione)
Preparazioni	(<u>IdSessionePratica</u> , <u>IdRicetta</u>)
Utilizzi	(<u>IdRicetta</u> , <u>IdIngrediente</u> , Quantità, UDM)

Table 3: Schema logico delle entità

4.1 Vincoli Referenziali

Entità	Chiavi esterne
Corso	<u>IdChef</u> → Chef(<u>IdChef</u>)
SessionePratica	<u>IdCorso</u> → Corso(<u>IdCorso</u>)
SessioneOnline	<u>IdCorso</u> → Corso(<u>IdCorso</u>)
Ricetta	<u>IdChef</u> → Chef(<u>IdChef</u>)
Iscrizioni	<u>IdPartecipante</u> → Partecipante(<u>IdPartecipante</u>) <u>IdCorso</u> → Corso(<u>IdCorso</u>)
Argomenti_Corso	<u>IdCorso</u> → Corso(<u>IdCorso</u>) <u>IdArgomento</u> → Argomento(<u>IdArgomento</u>)
Adesioni	<u>IdPartecipante</u> → Partecipante(<u>IdPartecipante</u>) <u>IdSessionePratica</u> → SessionePratica(<u>IdSessionePratica</u>)
Preparazioni	<u>IdSessionePratica</u> → SessionePratica(<u>IdSessionePratica</u>) <u>IdRicetta</u> → Ricetta(<u>IdRicetta</u>)
Utilizzi	<u>IdRicetta</u> → Ricetta(<u>IdRicetta</u>) <u>IdIngrediente</u> → Ingrediente(<u>IdIngrediente</u>)

Table 4: Vincoli referenziali

5 Schema fisico

5.1 Struttura SQL delle tabelle

```
1  -- 00_Create: Definizione Tabelle
2
3  -- Partecipante
4  CREATE TABLE Partecipante (
5      IdPartecipante SERIAL PRIMARY KEY,
6      Username TEXT UNIQUE NOT NULL,
7      Nome VARCHAR(100) NOT NULL,
8      Cognome VARCHAR(100) NOT NULL,
9      CodiceFiscale CHAR(16) UNIQUE NOT NULL,
10     DataDiNascita DATE NOT NULL,
11     LuogoDiNascita VARCHAR(100) NOT NULL,
12     Email TEXT UNIQUE NOT NULL,
13     Password VARCHAR(60) NOT NULL,
14     CONSTRAINT check_empty_part_nome CHECK (LENGTH(Nome) > 0),
15     CONSTRAINT check_empty_part_cognome CHECK (LENGTH(Cognome) > 0),
16     CONSTRAINT check_empty_part_cf CHECK (LENGTH(CodiceFiscale) = 16),
17     CONSTRAINT check_empty_part_luogonascita CHECK (LENGTH(LuogoDiNascita) > 0),
18     CONSTRAINT check_empty_part_email CHECK (LENGTH(Email) > 0),
19     CONSTRAINT check_empty_part_pass CHECK (LENGTH>Password) > 0),
20     CONSTRAINT check_part_length CHECK (LENGTH(Username) BETWEEN 4 AND 20),
21     CONSTRAINT check_part_maggiorenne CHECK (DataDiNascita <= CURRENT_DATE -
        INTERVAL '18 years')
22 );
23
24 -- Chef
25 CREATE TABLE Chef (
26     IdChef SERIAL PRIMARY KEY,
27     Username TEXT UNIQUE NOT NULL,
28     Nome VARCHAR(100) NOT NULL,
29     Cognome VARCHAR(100) NOT NULL,
30     CodiceFiscale CHAR(16) UNIQUE NOT NULL,
31     DataDiNascita DATE NOT NULL,
32     LuogoDiNascita VARCHAR(100) NOT NULL,
33     Email TEXT UNIQUE NOT NULL,
34     Password VARCHAR(60) NOT NULL,
35     Curriculum TEXT NOT NULL,
36     CONSTRAINT check_empty_chef_nome CHECK (LENGTH(Nome) > 0),
37     CONSTRAINT check_empty_chef_cognome CHECK (LENGTH(Cognome) > 0),
38     CONSTRAINT check_correct_chef_cf CHECK (LENGTH(CodiceFiscale) = 16),
39     CONSTRAINT check_empty_chef_luogonascita CHECK (LENGTH(LuogoDiNascita) > 0),
40     CONSTRAINT check_empty_chef_email CHECK (LENGTH(Email) > 0),
41     CONSTRAINT check_empty_chef_pass CHECK (LENGTH>Password) > 0),
42     CONSTRAINT check_empty_chef_curriculum CHECK (LENGTH(Curriculum) > 0),
43     CONSTRAINT check_chef_length CHECK (LENGTH(Username) BETWEEN 4 AND 20),
44     CONSTRAINT check_chef_maggiorenne CHECK (DataDiNascita <= CURRENT_DATE -
        INTERVAL '18 years')
45 );
```



```

46
47 -- Argomento
48 CREATE TABLE Argomento (
49     IdArgomento SERIAL PRIMARY KEY,
50     Nome VARCHAR(100) UNIQUE NOT NULL
51 );
52
53 -- Enum per la frequenza delle sessioni
54 CREATE TYPE Frequenza AS ENUM (
55     'Giornaliera',
56     'Settimanale',
57     'Bisettimanale',
58     'Mensile',
59     'Libera'
60 );
61
62 -- Corso
63 CREATE TABLE Corso (
64     IdCorso SERIAL PRIMARY KEY,
65     Nome VARCHAR(100) NOT NULL,
66     DataInizio DATE NOT NULL,
67     NumeroSessioni INTEGER DEFAULT 0,
68     FrequenzaSessioni Frequenza NOT NULL,
69     Limite INTEGER DEFAULT NULL,
70     Descrizione TEXT NOT NULL,
71     Costo NUMERIC(10,2) NOT NULL,
72     isPratico BOOLEAN NOT NULL DEFAULT false,
73     IdChef INTEGER NOT NULL,
74     CONSTRAINT check_empty_corso_nome CHECK (LENGTH(Nome) > 0),
75     CONSTRAINT check_data_non_passata_corso CHECK (DataInizio > CURRENT_DATE),
76     CONSTRAINT check_empty_corso_descr CHECK (LENGTH(Descrizione) > 0),
77     CONSTRAINT check_costo_non_negativo CHECK (Costo >= 0.0),
78     CONSTRAINT check_limite_pratico CHECK (
79         (NOT isPratico AND LIMITE IS NULL) OR
80         (isPratico AND Limite IS NOT NULL)
81     ),
82     CONSTRAINT fk_chef_corso FOREIGN KEY(IdChef)
83         REFERENCES Chef(IdChef) ON DELETE CASCADE
84 );
85
86 -- SessionePratica
87 CREATE TABLE SessionePratica (
88     IdSessionePratica SERIAL PRIMARY KEY,
89     Durata INTEGER NOT NULL,
90     Orario TIME NOT NULL,
91     Data DATE NOT NULL,
92     NumeroPartecipanti INTEGER DEFAULT 0,
93     Luogo VARCHAR(100) NOT NULL,
94     IdCorso INTEGER NOT NULL,
95     CONSTRAINT check_data_non_passata_sessioneprat CHECK (Data > CURRENT_DATE),
96     CONSTRAINT check_durata_positiva_sessioneprat CHECK (Durata > 0),

```

```

97     CONSTRAINT check_empty_sessioneprat_luogo CHECK (LENGTH(Luogo) > 0),
98     CONSTRAINT fk_corso_pratica FOREIGN KEY(IdCorso)
99         REFERENCES Corso(IdCorso) ON DELETE CASCADE
100 );
101
102 -- SessioneOnline
103 CREATE TABLE SessioneOnline (
104     IdSessioneOnline SERIAL PRIMARY KEY,
105     Durata INTEGER NOT NULL,
106     Orario TIME NOT NULL,
107     Data DATE NOT NULL,
108     LinkRiunione TEXT NOT NULL,
109     IdCorso INTEGER NOT NULL,
110     CONSTRAINT check_data_non_passata_sessioneonl CHECK (Data > CURRENT_DATE),
111     CONSTRAINT check_durata_positiva_sessioneonl CHECK (Durata > 0),
112     CONSTRAINT check_empty_sessioneonl_link CHECK (LENGTH(LinkRiunione) > 0),
113     CONSTRAINT fk_corso_online FOREIGN KEY(IdCorso)
114         REFERENCES Corso(IdCorso) ON DELETE CASCADE
115 );
116
117 -- Enum per il tipo di difficulta della ricetta
118 CREATE TYPE LivelloDifficolta AS ENUM (
119     'Principiante',
120     'Facile',
121     'Medio',
122     'Difficile',
123     'Esperto'
124 );
125
126 -- Ricetta
127 CREATE TABLE Ricetta (
128     IdRicetta SERIAL PRIMARY KEY,
129     Nome VARCHAR(100) NOT NULL,
130     Provenienza VARCHAR(100) NOT NULL,
131     Tempo INTEGER NOT NULL,
132     Calorie INTEGER NOT NULL,
133     Difficolta LivelloDifficolta NOT NULL,
134     Allergeni VARCHAR(100),
135     IdChef INTEGER NOT NULL,
136     CONSTRAINT unique_nome_chef UNIQUE (Nome, IdChef),
137     CONSTRAINT check_empty_ricetta_nome CHECK (LENGTH(Nome) > 0),
138     CONSTRAINT check_empty_ricetta_provenienza CHECK (LENGTH(Provenienza) > 0),
139     CONSTRAINT check_tempo_positivo CHECK (Tempo > 0),
140     CONSTRAINT check_calorie_positive CHECK (Calorie > 0),
141     CONSTRAINT fk_chef_ricetta FOREIGN KEY(IdChef)
142         REFERENCES Chef(IdChef) ON DELETE CASCADE
143 );
144
145
146
147

```

```

148 -- Enum per l'origine dell'ingrediente
149 CREATE TYPE NaturaIngrediente AS ENUM (
150     'Vegetale',
151     'Animale',
152     'Minerale',
153     'Fungino',
154     'Sintetico',
155     'Microbico',
156     'Altro'
157 );
158
159 -- Ingrediente
160 CREATE TABLE Ingrediente (
161     IdIngrediente SERIAL PRIMARY KEY,
162     Nome VARCHAR(100) UNIQUE NOT NULL,
163     Origine NaturaIngrediente NOT NULL,
164     CONSTRAINT check_empty_ingr_nome CHECK (LENGTH(Nome) > 0)
165 );
166
167 -- Iscrizioni
168 CREATE TABLE Iscrizioni (
169     IdPartecipante INTEGER NOT NULL,
170     IdCorso INTEGER NOT NULL,
171     CONSTRAINT pk_iscrizioni_corso PRIMARY KEY(IdPartecipante, IdCorso),
172     CONSTRAINT fk_partecipante_iscrizioni FOREIGN KEY(IdPartecipante)
173         REFERENCES Partecipante(IdPartecipante) ON DELETE CASCADE,
174     CONSTRAINT fk_corso_iscrizioni FOREIGN KEY(IdCorso)
175         REFERENCES Corso(IdCorso) ON DELETE CASCADE
176 );
177
178 -- Argomenti_Corso
179 CREATE TABLE Argomenti_Corso (
180     IdCorso INTEGER NOT NULL,
181     IdArgomento INTEGER NOT NULL,
182     CONSTRAINT pk_argomenti_corso PRIMARY KEY(IdCorso, IdArgomento),
183     CONSTRAINT fk_corso_argomenticorso FOREIGN KEY(IdCorso)
184         REFERENCES Corso(IdCorso) ON DELETE CASCADE,
185     CONSTRAINT fk_argomenti_argomenticorso FOREIGN KEY(IdArgomento)
186         REFERENCES Argomento(IdArgomento) ON DELETE RESTRICT
187 );
188
189 -- Adesioni
190 CREATE TABLE Adesioni (
191     IdPartecipante INTEGER NOT NULL,
192     IdSessionePratica INTEGER NOT NULL,
193     DataAdesione DATE NOT NULL,
194     CONSTRAINT pk_adesioni PRIMARY KEY(IdPartecipante, IdSessionePratica),
195     CONSTRAINT fk_partecipante_adesioni FOREIGN KEY(IdPartecipante)
196         REFERENCES Partecipante(IdPartecipante),
197     CONSTRAINT fk_sessionepratica_adesioni FOREIGN KEY(IdSessionePratica)
198         REFERENCES SessionePratica(IdSessionePratica) ON DELETE CASCADE );

```

```

199 -- Preparazioni
200 CREATE TABLE Preparazioni (
201     IdSessionePratica INTEGER NOT NULL,
202     IdRicetta INTEGER NOT NULL,
203     CONSTRAINT pk_sessionepratica_ricetta PRIMARY KEY (IdSessionePratica,
204         IdRicetta),
205     CONSTRAINT fk_sessionepratica_preparazioni FOREIGN KEY(IdSessionePratica)
206         REFERENCES SessionePratica(IdSessionePratica) ON DELETE CASCADE,
207     CONSTRAINT fk_ricetta_preparazioni FOREIGN KEY(IdRicetta)
208         REFERENCES Ricetta(IdRicetta) ON DELETE CASCADE
209 );
210
211 -- Enumerazione per l'unita di misura degli utilizzi degli ingredienti
212 CREATE TYPE UnitaDiMisura AS ENUM (
213     'Unita',
214     'Grammi',
215     'Kilogrammi',
216     'Litri',
217     'Millilitri',
218     'Bicchiere',
219     'Tazza',
220     'Tazzina',
221     'Cucchiaino',
222     'Cucchiaio'
223 );
224
225 -- Utilizzi
226 CREATE TABLE Utilizzi (
227     IdRicetta INTEGER NOT NULL,
228     IdIngrediente INTEGER NOT NULL,
229     Quantita FLOAT8 NOT NULL,
230     UDM UnitaDiMisura NOT NULL,
231     CONSTRAINT pk_ricetta_ingrediente PRIMARY KEY(IdRicetta, IdIngrediente),
232     CONSTRAINT fk_ricetta_utilizzi FOREIGN KEY(IdRicetta)
233         REFERENCES Ricetta(IdRicetta) ON DELETE CASCADE,
234     CONSTRAINT fk_ingrediente_utilizzi FOREIGN KEY(IdIngrediente)
235         REFERENCES Ingrediente(IdIngrediente) ON DELETE RESTRICT,
236     CONSTRAINT check_quantita CHECK (Quantita > 0)
237 );

```

5.2 Trigger e procedure SQL

```
1  -- 01_Triggers_Functions_Procedures
2
3  -- Normalizzazione basica dei dati di Partecipante/Chef/Ricetta/Ingrediente
4
5  -- Funzione che normalizza i campi testuale dell'utente:
6  -- Username, Email -> Minuscolo
7  -- CodiceFiscale -> Maiuscolo
8  -- Nome, Cognome, Luogo -> iniziale maiuscola
9
10 CREATE OR REPLACE FUNCTION fun_normalizza_utente()
11 RETURNS TRIGGER AS
12 $$
13 BEGIN
14     NEW.Username := LOWER(NEW.Username);
15     NEW.Nome := INITCAP(NEW.Nome);
16     NEW.Cognome := INITCAP(NEW.Cognome);
17     NEW.CodiceFiscale := UPPER(NEW.CodiceFiscale);
18     NEW.LuogoDiNascita := INITCAP(NEW.LuogoDiNascita);
19     NEW.Email := LOWER(NEW.Email);
20
21     RETURN NEW;
22
23 END;
24 $$ LANGUAGE plpgsql;
25
26 -- Applica la normalizzazione prima di INSERT o UPDATE su Partecipante
27 CREATE TRIGGER trg_normalizza_partecipante
28 BEFORE INSERT OR UPDATE ON Partecipante
29 FOR EACH ROW
30 EXECUTE FUNCTION fun_normalizza_utente();
31
32 -- Applica la normalizzazione prima di INSERT o UPDATE su Chef
33 CREATE TRIGGER trg_normalizza_chef
34 BEFORE INSERT OR UPDATE ON Chef
35 FOR EACH ROW
36 EXECUTE FUNCTION fun_normalizza_utente();
37
38
39 -- Normalizzo Ingrediente e Ricetta con la prima lettera maiuscola
40
41 CREATE OR REPLACE FUNCTION fun_normalizza_ricetta_ingr_chef()
42 RETURNS TRIGGER AS
43 $$
44 BEGIN
45     NEW.Nome := INITCAP(NEW.Nome);
46     RETURN NEW;
47
48 END;
49 $$ LANGUAGE plpgsql;
```

```

50
51 -- Applica INITCAP al Nome di Ingrediente
52 CREATE TRIGGER trg_normalizza_ingrediente
53 BEFORE INSERT OR UPDATE ON Ingrediente
54 FOR EACH ROW
55 EXECUTE FUNCTION fun_normalizza_ricetta_ingr_chef();
56
57 -- Applica INITCAP al Nome di Ricetta
58 CREATE TRIGGER trg_normalizza_ricetta
59 BEFORE INSERT OR UPDATE ON Ricetta
60 FOR EACH ROW
61 EXECUTE FUNCTION fun_normalizza_ricetta_ingr_chef();
62
63 -----
64
65 -- Interrelazionale: Impedire che due utenti      uno Chef e uno Partecipante
66 -- abbiano lo stesso Username, anche se sono in due tabelle diverse.
67
68 CREATE OR REPLACE FUNCTION fun_username_unique()
69 RETURNS TRIGGER AS
70 $$
71 BEGIN
72     -- Se      un UPDATE e l' username non cambia, salto il controllo
73     IF TG_OP = 'UPDATE' AND NEW.Username = OLD.Username THEN
74
75         RETURN NEW;
76     END IF;
77
78     -- Se stiamo inserendo o modificando un Partecipante, controlla che l'
79     -- username non sia gi  usato da uno CHEF
80     IF TG_TABLE_NAME = 'partecipante' THEN
81         IF EXISTS (SELECT 1 FROM Chef WHERE Username = NEW.Username) THEN
82             RAISE EXCEPTION 'Username gi  usato in Chef';
83         END IF;
84     END IF;
85
86     -- Se stiamo inserendo o modificando uno Chef, controlla che l'username
87     -- non sia gi  usato da un Partecipante
88     IF TG_TABLE_NAME = 'chef' THEN
89         IF EXISTS (SELECT 1 FROM Partecipante WHERE Username = NEW.
90             Username) THEN
91             RAISE EXCEPTION 'Username gi  usato in Partecipante';
92         END IF;
93     END IF;
94     RETURN NEW;
95 END;
96 $$ LANGUAGE plpgsql;
97
98 -- Trigger che garantiscono unicit  dello username tra le due tabelle (vincolo
99 -- nato dalla ristrutturazione)

```

```

96 CREATE TRIGGER trg_unico_username_partecipante
97 BEFORE INSERT OR UPDATE ON Partecipante
98 FOR EACH ROW
99 EXECUTE FUNCTION fun_username_unique();
100
101 CREATE TRIGGER trg_unico_username_chef
102 BEFORE INSERT OR UPDATE ON Chef
103 FOR EACH ROW
104 EXECUTE FUNCTION fun_username_unique();
105
106 -----
107
108 -- Gestione numero sessioni del corso automatica
109
110 -- Blocca l'inserimento di un corso con NumeroSessioni diverso da 0, il numero
    gestito dal db automaticamente
111
112 CREATE OR REPLACE FUNCTION fun_setta_numero_sessioni_corsi_iniziali()
113 RETURNS TRIGGER AS
114 $$
115 BEGIN
116     IF NEW.NumeroSessioni <> 0 THEN
117         RAISE EXCEPTION 'Il corso deve partire con numero di
    sessioni 0! Vengono aggiornate automaticamente';
118     END IF;
119     RETURN NEW;
120 END;
121 $$ LANGUAGE plpgsql;
122
123 CREATE TRIGGER trg_setta_numero_sessioni_corsi_iniziali
124 BEFORE INSERT ON Corso
125 FOR EACH ROW
126 EXECUTE FUNCTION fun_setta_numero_sessioni_corsi_iniziali();
127
128
129 -- Aggiornamento numero sessioni (incremento quando viene creata una nuova
    sessione di qualsiasi tipo)
130
131 CREATE OR REPLACE FUNCTION fun_incrementa_num_sessioni()
132 RETURNS TRIGGER AS
133 $$
134 BEGIN
135     UPDATE Corso
136     SET NumeroSessioni = NumeroSessioni + 1
137     WHERE IdCorso = NEW.IdCorso;
138     RETURN NULL;
139 END;
140 $$ LANGUAGE plpgsql;
141
142
143

```

```

144 -- Incremento automatico all'inserimento di una sessione (Online/Pratica)
145 CREATE TRIGGER trg_incremento_numsessioni_online
146 AFTER INSERT ON SessioneOnline
147 FOR EACH ROW
148 EXECUTE FUNCTION fun_incrementa_num_sessioni();
149
150 CREATE TRIGGER trg_incremento_numsessioni_pratiche
151 AFTER INSERT ON SessionePratica
152 FOR EACH ROW
153 EXECUTE FUNCTION fun_incrementa_num_sessioni();
154
155
156 -- Aggiornamento numero sessioni (decremento quando viene eliminata una nuova
    sessione di qualsiasi tipo)
157 -- Protezione con GREATEST per evitare valori negativi
158
159 CREATE OR REPLACE FUNCTION fun_decrementa_num_sessioni()
160 RETURNS TRIGGER AS
161 $$
162 BEGIN
163     IF EXISTS (SELECT 1 FROM Corso WHERE IdCorso = OLD.IdCorso) THEN
164         UPDATE Corso
165         SET NumeroSessioni = GREATEST(NumeroSessioni - 1, 0)
166         WHERE IdCorso = OLD.IdCorso;
167     END IF;
168     RETURN OLD;
169 END;
170 $$ LANGUAGE plpgsql;
171
172
173 -- Decremento automatico alla rimozione di una sessione (Online/Pratica)
174 CREATE TRIGGER trg_decrementa_num_sessioni_pratica
175 AFTER DELETE ON SessionePratica
176 FOR EACH ROW
177 EXECUTE FUNCTION fun_decrementa_num_sessioni();
178
179 CREATE TRIGGER trg_decrementa_num_sessioni_online
180 AFTER DELETE ON SessioneOnline
181 FOR EACH ROW
182 EXECUTE FUNCTION fun_decrementa_num_sessioni();
183
184
185
186
187
188
189
190
191
192
193 -----

```



```

194 -- Gestione sessioni pratiche del corso automatica
195
196 -- Blocca l'inserimento di una sessione pratica se il corso non    pratico
197 --(poteva essere impostato automaticamente a true all'inserimento di una sessione
    pratica, ma non sarebbe stato possibile inserire il limite corrispondente)
198
199 CREATE OR REPLACE FUNCTION fun_ispratico_insert()
200 RETURNS TRIGGER AS
201 $$
202 DECLARE
203     pratico Corso.isPratico%TYPE;
204 BEGIN
205     SELECT isPratico INTO pratico FROM Corso WHERE IdCorso = NEW.IdCorso;
206
207     IF NOT pratico THEN
208         RAISE EXCEPTION 'Non puoi inserire una sessione pratica in un
            corso non pratico!!!';
209     END IF;
210     RETURN NEW;
211
212 END;
213 $$ LANGUAGE plpgsql;
214
215 CREATE TRIGGER trg_ispratico_insert
216 BEFORE INSERT ON SessionePratica
217 FOR EACH ROW
218 EXECUTE FUNCTION fun_ispratico_insert();
219
220
221 -- Blocca l'inserimento di una sessione con NumeroPartecipanti diverso da 0, il
    numero    gestito dal db automaticamente
222
223 CREATE OR REPLACE FUNCTION fun_setta_numero_partecipanti_iniziali()
224 RETURNS TRIGGER AS
225 $$
226 BEGIN
227     IF NEW.NumeroPartecipanti <> 0 THEN
228         RAISE EXCEPTION 'La Sessione pratica deve partire con
            numero di partecipanti 0! Vengono aggiornati
            automaticamente';
229     END IF;
230     RETURN NEW;
231 END;
232 $$ LANGUAGE plpgsql;
233
234 CREATE TRIGGER trg_setta_numero_partecipanti_iniziali
235 BEFORE INSERT ON SessionePratica
236 FOR EACH ROW
237 EXECUTE FUNCTION fun_setta_numero_partecipanti_iniziali();
238
239

```

```

240 -- Verifica che l'adesione sia registrata alla data corrente e previene
      inserimenti retrodatati o futuri
241
242 CREATE OR REPLACE FUNCTION fun_setta_data_adesione()
243 RETURNS TRIGGER AS
244 $$
245 BEGIN
246     IF NEW.DataAdesione <> CURRENT_DATE THEN
247         RAISE EXCEPTION 'Se inserisci una adesione alla sessione pratica
              allora la sua data deve essere quella di oggi';
248     END IF;
249     RETURN NEW;
250 END;
251 $$ LANGUAGE plpgsql;
252
253 CREATE TRIGGER trg_setta_data_adesione
254 BEFORE INSERT ON Adesioni
255 FOR EACH ROW
256 EXECUTE FUNCTION fun_setta_data_adesione();
257
258
259 -- Incrementa il numero partecipanti di una sessione pratica al momento dell'
      adesione di un partecipante
260
261 CREATE OR REPLACE FUNCTION fun_incrementa_num_utenti()
262 RETURNS TRIGGER AS
263 $$
264 BEGIN
265     UPDATE SessionePratica
266     SET NumeroPartecipanti = NumeroPartecipanti + 1
267     WHERE IdSessionePratica = NEW.IdSessionePratica;
268     RETURN NULL;
269 END;
270 $$ LANGUAGE plpgsql;
271
272 CREATE TRIGGER trg_incremento_numutenti
273 AFTER INSERT ON Adesioni
274 FOR EACH ROW
275 EXECUTE FUNCTION fun_incrementa_num_utenti();
276
277
278 -- Decrementa il numero partecipanti di una sessione pratica al momento dell'
      adesione di un partecipante con protezione per non andare in negativo
279
280 CREATE OR REPLACE FUNCTION fun_decrementa_num_utenti()
281 RETURNS TRIGGER AS
282 $$
283 BEGIN
284     UPDATE SessionePratica
285     SET NumeroPartecipanti = GREATEST(NumeroPartecipanti - 1, 0)
286     WHERE IdSessionePratica = OLD.IdSessionePratica;

```

```

287         RETURN OLD;
288 END;
289 $$ LANGUAGE plpgsql;
290
291 CREATE TRIGGER trg_decrementa_num_utenti
292 AFTER DELETE ON Adesioni
293 FOR EACH ROW
294 EXECUTE FUNCTION fun_decrementa_num_utenti();
295
296
297 -----
298 -- Gestione sessioni (frequenza e orari)
299
300 -- Impedisce pi sessioni (pratiche o online) dello stesso corso nello stesso
    giorno
301 -- Controlla entrambe le tabelle SessionePratica e SessioneOnline per evitare
    duplicati o sovrapposizioni
302 -- Escludiamo i conflitti (Potremmo considerare la sessione stessa in update e
    EXISTS ci restituirebbe true)
303
304 CREATE OR REPLACE FUNCTION fun_unicita_sessione_giorno()
305 RETURNS TRIGGER AS
306 $$
307 BEGIN
308
309     IF TG_TABLE_NAME = 'sessionepratica' THEN
310         IF EXISTS
311             (
312                 -- Verifica se esiste gi una sessione pratica con
313                 stessa data e corso, diversa dalla riga in update
314                 SELECT 1 FROM SessionePratica
315                 WHERE Data = NEW.Data
316                 AND IdCorso = NEW.IdCorso
317                 AND NOT (TG_OP = 'UPDATE' AND IdSessionePratica = NEW.
318                     IdSessionePratica)
319             )
320         THEN RAISE EXCEPTION 'Esiste gi una sessione pratica
321             per questo corso in data %.', NEW.Data;
322         END IF;
323     IF EXISTS
324         (
325             -- Verifica se esiste gi una sessione online con
326             stessa data e corso
327             SELECT 1 FROM SessioneOnline
328             WHERE Data = NEW.Data
329             AND IdCorso = NEW.IdCorso
330         )
331     THEN RAISE EXCEPTION 'Esiste gi una sessione online per
332         questo corso in data %.', NEW.Data;
333     END IF;
334 ELSE
335     IF EXISTS

```

```

329         (          -- Verifica se esiste gi una sessione online
330                     con stessa data e corso, diversa dalla riga in update
331 SELECT 1 FROM SessioneOnline
332 WHERE Data = NEW.Data
333 AND IdCorso = NEW.IdCorso
334 AND NOT (TG_OP = 'UPDATE' AND IdSessioneOnline = NEW.
        IdSessioneOnline)
335 )
336 THEN RAISE EXCEPTION 'Esiste gi una sessione online per
        questo corso in data %.', NEW.Data;
337 END IF;
338 IF EXISTS
339 (          -- Verifica se esiste gi una sessione pratica
340                     con stessa data e corso
341 SELECT 1 FROM SessionePratica
342 WHERE Data = NEW.Data
343 AND IdCorso = NEW.IdCorso
344 )
345 THEN RAISE EXCEPTION 'Esiste gi una sessione pratica
        per questo corso in data %.', NEW.Data;
346 END IF;
347 END IF;
348 RETURN NEW;
349 END;
350 $$ LANGUAGE plpgsql;
351
352 CREATE TRIGGER trg_unicita_sessione_online_giorno
353 BEFORE INSERT OR UPDATE OF Data ON SessioneOnline
354 FOR EACH ROW
355 EXECUTE FUNCTION fun_unicita_sessione_giorno();
356
357 CREATE TRIGGER trg_unicita_sessione_pratica_giorno
358 BEFORE INSERT OR UPDATE OF Data ON SessionePratica
359 FOR EACH ROW
360 EXECUTE FUNCTION fun_unicita_sessione_giorno();
361
362 -- Controlla che la data della sessione:
363 -- 1- Non sia precedente alla data di inizio del corso
364 -- 2- Se la prima sessione di quel corso, deve coincidere con la data di
        inizio corso
365
366 CREATE OR REPLACE FUNCTION fun_sessione_dopo_inizio_corso()
367 RETURNS TRIGGER AS
368 $$
369 DECLARE
370     data_inizio Corso.DataInizio%TYPE;
371 BEGIN
372     SELECT DataInizio INTO data_inizio FROM Corso WHERE IdCorso = NEW.IdCorso
        ;

```

```

373
374     IF NEW.Data < data_inizio THEN
375         RAISE EXCEPTION 'La sessione non pu essere precedente alla data
           di inizio del corso';
376     END IF;
377
378     -- Se non esistono sessioni per il corso, la prima deve essere
           esattamente la data di inizio corso
379     IF NOT EXISTS
380         (
381             SELECT 1 FROM SessionePratica WHERE IdCorso =
                 NEW.IdCorso
382             UNION
383             SELECT 1 FROM SessioneOnline WHERE IdCorso =
                 NEW.IdCorso
384         )
385     THEN
386         IF NEW.Data <> data_inizio THEN
387             RAISE EXCEPTION 'La prima sessione del corso deve essere
                 il giorno della Data di inizio del corso';
388         END IF;
389     END IF;
390
391     RETURN NEW;
392 END;
393 $$ LANGUAGE plpgsql;
394
395 CREATE TRIGGER trg_sessione_pratica_dopo_inizio_corso
396 BEFORE INSERT ON SessionePratica
397 FOR EACH ROW
398 EXECUTE FUNCTION fun_sessione_dopo_inizio_corso();
399
400 CREATE TRIGGER trg_sessione_online_dopo_inizio_corso
401 BEFORE INSERT ON SessioneOnline
402 FOR EACH ROW
403 EXECUTE FUNCTION fun_sessione_dopo_inizio_corso();
404
405
406 -- Verifica della frequenza all'inserimento o aggiornamento della data di una
           sessione
407
408 CREATE OR REPLACE FUNCTION fun_verifica_frequenza_sessioni()
409 RETURNS TRIGGER AS
410 $$
411 DECLARE
412     frequenza Corso.FrequenzaSessioni%TYPE; -- Frequenza del corso
413     giorniFrequenza INTEGER;                 -- Numero di giorni che corrisponde
           alla frequenza.
414     v_dataInizio DATE;                       -- Data di inizio del corso
415     finestra_nuova INTEGER;                  -- Indice della finestra temporale
           per la nuova data.

```

```

416     finestra_vecchia INTEGER;                -- Indice della finestra temporale
        per la vecchia data (UPDATE).
417     id_sessione_corrente INTEGER;            -- ID della sessione in corso di
        modifica (per auto-esclusione).
418 BEGIN
419     -- Recupera i dati del corso
420     SELECT FrequenzaSessioni, DataInizio
421     INTO frequenza, v_dataInizio
422     FROM Corso
423     WHERE IdCorso = NEW.IdCorso;
424
425     -- La data della sessione non pu essere antecedente alla data di inizio del
        corso
426     IF NEW.Data < v_dataInizio THEN
427         RAISE EXCEPTION 'La data della sessione (%) non pu essere precedente
        alla data di inizio del corso (%).', NEW.Data, v_dataInizio;
428     END IF;
429
430     -- Se la frequenza 'Libera', non sono necessari ulteriori controlli
431     IF frequenza = 'Libera' THEN
432         RETURN NEW;
433     END IF;
434
435     -- Converte la frequenza testuale in un numero di giorni per il calcolo delle
        finestre temporali
436     CASE frequenza
437         WHEN 'Giornaliera' THEN giorniFrequenza := 1;
438         WHEN 'Settimanale' THEN giorniFrequenza := 7;
439         WHEN 'Bisettimanale' THEN giorniFrequenza := 14;
440         WHEN 'Mensile' THEN giorniFrequenza := 30;
441         ELSE
442             RAISE EXCEPTION 'Frequenza non riconosciuta: %', frequenza;
443     END CASE;
444
445     -- Calcola l'indice della finestra per la nuova data della sessione.
446     -- (DataSessione - DataInizioCorso) / 7 giorni -> 0 per la prima settimana, 1
        per la seconda, ...
447
448     finestra_nuova := FLOOR((NEW.Data - v_dataInizio) / giorniFrequenza);
449
450     -- Se la sessione viene modificata ma rimane nella stessa finestra temporale,
451     -- l'operazione valida e non sono necessari altri controlli
452     IF TG_OP = 'UPDATE' THEN
453         finestra_vecchia := FLOOR((OLD.Data - v_dataInizio) / giorniFrequenza);
454         IF finestra_nuova = finestra_vecchia THEN
455             RETURN NEW;
456         END IF;
457     END IF;
458
459     -- Identifica la sessione corrente per escluderla dal conflitto durante un
        UPDATE

```

```

460 IF TG_TABLE_NAME = 'sessionepratica' THEN
461     id_sessione_corrente := NEW.IdSessionePratica;
462 ELSE
463     id_sessione_corrente := NEW.IdSessioneOnline;
464 END IF;
465
466 -- Verifica se esiste gi un'altra sessione nella finestra temporale di
    destinazione
467 IF EXISTS
468 (
469     -- Unifico le sessioni
470     SELECT 1
471     FROM (
472         SELECT Data, IdSessionePratica AS id, 'SessionePratica' as tipo
473             FROM SessionePratica WHERE IdCorso = NEW.IdCorso
474         UNION ALL
475         SELECT Data, IdSessioneOnline AS id, 'SessioneOnline' as tipo
476             FROM SessioneOnline WHERE IdCorso = NEW.IdCorso
477     ) AS tutte_le_sessioni
478
479     -- Trovo le sessioni che cadono nella stessa finestra temporale calcolata
480     WHERE FLOOR((tutte_le_sessioni.Data - v_dataInizio) / giorniFrequenza) =
481         finestra_nuova
482
483     -- Esclude la sessione stessa dal controllo in caso di UPDATE
484     AND NOT (TG_OP = 'UPDATE' AND tutte_le_sessioni.id = id_sessione_corrente
485             AND tutte_le_sessioni.tipo = TG_TABLE_NAME)
486 )
487 THEN
488     RAISE EXCEPTION 'La finestra temporale per la data % gi occupata da
489         un''altra sessione.', NEW.Data;
490 END IF;
491 RETURN NEW;
492 END;
493 $$ LANGUAGE plpgsql;
494
495 CREATE TRIGGER trg_verifica_frequenza_sessioni_pratiche
496 BEFORE INSERT OR UPDATE OF Data ON SessionePratica
497 FOR EACH ROW
498 EXECUTE FUNCTION fun_verifica_frequenza_sessioni();
499
500 CREATE TRIGGER trg_verifica_frequenza_sessioni_online
501 BEFORE INSERT OR UPDATE OF Data ON SessioneOnline
502 FOR EACH ROW
503 EXECUTE FUNCTION fun_verifica_frequenza_sessioni();
504
505 -- Se una sessione associata al corso viene cancellata, impostiamo la frequenza
    del corso a libera
506 --(questo perch vogliamo mantenere un controllo della frequenza senza
    violazioni ed evitare buchi)

```

```

503
504 CREATE OR REPLACE FUNCTION fun_gestisci_frequenza_dopo_eliminazione()
505 RETURNS TRIGGER AS
506 $$
507 BEGIN
508     -- Controlla se la frequenza del corso non gi 'Libera'
509     -- per evitare update ridondante
510     IF (SELECT FrequenzaSessioni FROM Corso WHERE IdCorso = OLD.IdCorso) <> '
511         Libera' THEN
512         UPDATE Corso
513         SET FrequenzaSessioni = 'Libera'
514         WHERE IdCorso = OLD.IdCorso;
515     END IF;
516     RETURN OLD;
517 END;
518 $$ LANGUAGE plpgsql;
519
520 CREATE TRIGGER trg_gestisci_frequenza_dopo_eliminazione_pratica
521 AFTER DELETE ON SessionePratica
522 FOR EACH ROW
523 EXECUTE FUNCTION fun_gestisci_frequenza_dopo_eliminazione();
524
525
526 CREATE TRIGGER trg_gestisci_frequenza_dopo_eliminazione_online
527 AFTER DELETE ON SessioneOnline
528 FOR EACH ROW
529 EXECUTE FUNCTION fun_gestisci_frequenza_dopo_eliminazione();
530
531
532 -- Controlliamo, all'inserimento o modifica di una sessione, che non ci siano
533     altre sessioni (di altri corsi) che siano in sovrapposizione temporale
534
535 CREATE OR REPLACE FUNCTION fun_controlla_sovrapposizione_orario_sessione()
536 RETURNS TRIGGER AS
537 $$
538 DECLARE
539     -- Recupera l'id dello chef del corso
540     chef Chef.IdChef%TYPE;
541
542     -- Calcola orario di inizio e fine della nuova sessione
543     inizio_nuova TIME := NEW.Orario;
544     fine_nuova TIME := (NEW.Orario + (NEW.Durata || ' minutes')::interval)::time;
545
546     -- Memorizza id e tipo della sessione corrente (pratica o online)
547     id_sessione_corrente INTEGER;
548     sessione_corrente TEXT;
549 BEGIN
550     -- Determina il tipo di sessione in base alla tabella
551     IF TG_TABLE_NAME = 'sessionepratica' THEN

```



```

552         id_sessione_corrente := NEW.IdSessionePratica;
553         sessione_corrente := 'sessionepratica';
554     ELSIF TG_TABLE_NAME = 'sessioneonline' THEN
555         id_sessione_corrente := NEW.IdSessioneOnline;
556         sessione_corrente := 'sessioneonline';
557     END IF;
558
559     -- Recupera lo chef del corso
560     SELECT IdChef INTO chef FROM Corso WHERE IdCorso = NEW.IdCorso;
561
562     -- Controlla se esiste una sessione dello stesso chef, nello stesso
563     -- giorno, con orario sovrapposto
564     IF EXISTS
565     (
566         SELECT 1 FROM
567         (
568             -- Sessioni pratiche
569             SELECT SP.Data, SP.Orario, SP.Durata, C.IdChef, SP.
570             IdSessionePratica AS IdSessione, 'sessionepratica' AS
571             TipoSessione
572             FROM SessionePratica SP JOIN Corso C ON SP.IdCorso = C.IdCorso
573             WHERE C.IdChef = chef AND SP.Data = NEW.Data
574
575             UNION ALL
576
577             -- Sessioni online
578             SELECT SO.Data, SO.Orario, SO.Durata, C.IdChef, SO.
579             IdSessioneOnline AS IdSessione, 'sessioneonline' AS
580             TipoSessione
581             FROM SessioneOnline SO JOIN Corso C ON SO.IdCorso = C.IdCorso
582             WHERE C.IdChef = chef AND SO.Data = NEW.Data
583         ) AS S
584
585         -- Esclude la sessione stessa nel caso di UPDATE
586         WHERE NOT (TG_OP = 'UPDATE' AND S.TipoSessione = sessione_corrente
587         AND S.IdSessione = id_sessione_corrente)
588
589         -- Verifica sovrapposizione oraria: inizio < fine
590         -- esistente e fine > inizio esistente
591         AND NOT ((S.Orario + (S.Durata || ' minutes')::interval)
592         ::time <= inizio_nuova OR S.Orario >= fine_nuova)
593     ) THEN
594         RAISE EXCEPTION 'Sovrapposizione oraria con altra sessione dello stesso
595         chef';
596     END IF;
597     RETURN NEW;
598 END;
599 $$ LANGUAGE plpgsql;
600
601 CREATE TRIGGER trg_sovrapposizione_orario_sessione_pratica
602 BEFORE INSERT OR UPDATE OF Data, Orario, Durata ON SessionePratica

```

```

594 FOR EACH ROW
595 EXECUTE FUNCTION fun_controlla_sovrapposizione_orario_sessione();
596
597 CREATE TRIGGER trg_sovrapposizione_orario_sessione_online
598 BEFORE INSERT OR UPDATE OF Data, Orario, Durata ON SessioneOnline
599 FOR EACH ROW
600 EXECUTE FUNCTION fun_controlla_sovrapposizione_orario_sessione();
601
602 -----
603
604 -- Controlla all'insert sulla tabella ponte che gli argomenti del corso non siano
        essere pi di 5
605
606 CREATE OR REPLACE FUNCTION fun_limit_argomenti()
607 RETURNS TRIGGER AS
608 $$
609 DECLARE
610     num_argomenti INTEGER;
611 BEGIN
612
613     SELECT COUNT(*) INTO num_argomenti FROM Argomenti_Corso WHERE IdCorso =
        NEW.IdCorso;
614     IF num_argomenti >= 5 THEN
615         RAISE EXCEPTION 'E'' gia'' stato scelto il numero massimo di
            argomenti per questo corso';
616     END IF;
617     RETURN NEW;
618
619 END;
620 $$ LANGUAGE plpgsql;
621
622 CREATE TRIGGER trg_limit_argomenti
623 BEFORE INSERT OR UPDATE ON Argomenti_Corso
624 FOR EACH ROW
625 EXECUTE FUNCTION fun_limit_argomenti();
626
627 -----
628
629 -- Controlla che se viene inserita una iscrizione ma il limite di iscrizioni
        gi raggiunto, essa non viene inserita
630
631 CREATE OR REPLACE FUNCTION fun_limite_iscrizioni()
632 RETURNS TRIGGER AS
633 $$
634 DECLARE
635     numero_iscritti INTEGER;
636     limite_corso Corso.Limite%TYPE;
637     pratico Corso.isPratico%TYPE;
638 BEGIN
639
640     -- Ottiene il limite e il tipo del corso associato all'iscrizione

```

```

641         SELECT Limite, isPratico INTO limite_corso, pratico
642     FROM Corso
643     WHERE IdCorso = NEW.IdCorso;
644
645     -- Il controllo viene fatto solo per i corsi pratici (gli unici che hanno
        il limite)
646     IF pratico THEN
647         -- Conta quante iscrizioni esistono gi per quel corso
648         SELECT COUNT(*) INTO numero_iscritti
649         FROM Iscrizioni
650         WHERE IdCorso = NEW.IdCorso;
651
652         -- Se stato raggiunto il limite massimo, blocca l'
        inserimento
653         IF numero_iscritti >= limite_corso THEN
654             RAISE EXCEPTION 'Il limite delle iscrizioni per il corso
                stato gi raggiunto';
655         END IF;
656     END IF;
657     RETURN NEW;
658
659 END;
660 $$ LANGUAGE plpgsql;
661
662 CREATE TRIGGER trg_limite_iscrizioni
663 BEFORE INSERT ON Iscrizioni
664 FOR EACH ROW
665 EXECUTE FUNCTION fun_limite_iscrizioni();
666
667 -----
668
669 -- Un utente non puo' partecipare a una sessione pratica se non iscritto al corso
        che la organizza
670
671 CREATE OR REPLACE FUNCTION fun_iscrizione_before_adesione()
672 RETURNS TRIGGER AS
673 $$
674 DECLARE
675     idcorso_sessione SessionePratica.IdCorso%TYPE;
676 BEGIN
677
678     -- Recupero l'id del corso dalla sessionepratica associata
679     SELECT IdCorso INTO idcorso_sessione
680     FROM SessionePratica
681     WHERE IdSessionePratica = NEW.IdSessionePratica;
682
683     -- Se non esiste l'iscrizione del partecipante per quel corso,
        blocca l'iscrizione
684     IF NOT EXISTS ( SELECT 1 FROM Iscrizioni WHERE IdPartecipante =
        NEW.IdPartecipante AND IdCorso = idcorso_sessione ) THEN

```

```

685         RAISE EXCEPTION 'Partecipante non iscritto al corso, impossibile
        aderire alla sessione pratica';
686     END IF;
687     RETURN NEW;
688
689 END;
690 $$ LANGUAGE plpgsql;
691
692 CREATE TRIGGER trg_iscrizione_before_adesione
693 BEFORE INSERT ON Adesioni
694 FOR EACH ROW
695 EXECUTE FUNCTION fun_iscrizione_before_adesione();
696
697
698 -- Interrelazionale: La data dell'adesione alla sessione pratica deve essere
        antecedente (di almeno 3 giorni) alla data della sessione pratica.
699 -- Se la sessione      troppo vicina o gi      avvenuta, l'adesione viene bloccata.
700
701 CREATE OR REPLACE FUNCTION fun_data_adesione()
702 RETURNS TRIGGER AS
703 $$
704 DECLARE
705     Data_Sessione SessionePratica.Data%TYPE;
706 BEGIN
707     -- Recupero la data della sessione pratica
708     SELECT Data INTO Data_Sessione
709     FROM SessionePratica
710     WHERE IdSessionePratica = NEW.IdSessionePratica;
711
712     -- Se mancano meno di 3 giorni tra adesione e la data della
        sessione, blocca l'adesione
713     IF (Data_Sessione - NEW.DataAdesione < 3) THEN
714         RAISE EXCEPTION 'L'' adesione deve essere antecedente (3 giorni)
        alla data della sessione pratica';
715     END IF;
716
717     RETURN NEW;
718 END;
719 $$ LANGUAGE plpgsql;
720
721 CREATE TRIGGER trg_data_adesione
722 BEFORE INSERT ON Adesioni
723 FOR EACH ROW
724 EXECUTE FUNCTION fun_data_adesione();
725
726 -----
727
728 -- Verifica che lo chef della ricetta sia lo stesso chef che organizza il corso
        della sessione pratica.
729 -- Impedisce che ricette di altri chef vengano assegnate a sessioni non loro.
730

```

```

731 CREATE OR REPLACE FUNCTION fun_check_corso_chef()
732 RETURNS TRIGGER AS
733 $$
734 DECLARE
735     ChefRicetta Chef.IdChef%TYPE;
736     ChefCorso Chef.IdChef%TYPE;
737 BEGIN
738
739     -- Ottengo lo chef associato alla ricetta
740     SELECT IdChef
741     INTO ChefRicetta
742     FROM Ricetta R
743     WHERE R.IdRicetta = NEW.IdRicetta;
744
745     -- Ottengo lo chef associato al corso della sessione pratica
746     SELECT IdChef
747     INTO ChefCorso
748     FROM SessionePratica SP JOIN Corso C ON SP.IdCorso = C.IdCorso
749     WHERE SP.IdSessionePratica = NEW.IdSessionePratica;
750
751     -- Confronta i due chef e se sono diversi blocca l'inserimento della
752     -- ricetta nella sessione
753     IF ChefRicetta <> ChefCorso
754     THEN RAISE EXCEPTION 'Lo chef della ricetta e lo chef che organizza il
755         corso non sono gli stessi!';
756     END IF;
757
758     RETURN NEW;
759 END;
760 $$ LANGUAGE plpgsql;
761
762 CREATE TRIGGER trg_check_corso_chef
763 BEFORE INSERT ON Preparazioni
764 FOR EACH ROW
765 EXECUTE FUNCTION fun_check_corso_chef();
766
767 -----
768 -- Blocco di update di attributi e delete su tabelle
769 -----
770
771 -- Blocca la modifica del codice fiscale una volta che stato inserito
772
773 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_utente()
774 RETURNS TRIGGER AS
775 $$
776 BEGIN
777     RAISE EXCEPTION 'Non puoi modificare il tuo codice fiscale!!!';
778 END;
779 $$ LANGUAGE plpgsql;

```

```

780
781 CREATE TRIGGER trg_blocca_aggiorna_chef
782 BEFORE UPDATE OF CodiceFiscale ON Chef
783 FOR EACH ROW
784 EXECUTE FUNCTION fun_blocca_aggiorna_utente();
785
786 CREATE TRIGGER trg_blocca_aggiorna_partecipante
787 BEFORE UPDATE OF CodiceFiscale ON Partecipante
788 FOR EACH ROW
789 EXECUTE FUNCTION fun_blocca_aggiorna_utente();
790
791
792 -----
793
794 -- Impedisce l'eliminazione di un partecipante se ha aderito a una sessione
795 -- pratica imminente (entro 3 giorni).
796
797 -- Protegge la coerenza delle adesioni poco prima della sessione e impedisce gli
798 -- sprechi.
799
800 CREATE OR REPLACE FUNCTION fun_delete_partecipante_con_adesione()
801 RETURNS TRIGGER AS
802 $$
803 BEGIN
804     -- Controlla se esiste una adesione del partecipante che dista a meno di
805     -- 3 giorni dalla data di oggi
806     IF EXISTS(SELECT 1
807               FROM Adesioni NATURAL JOIN SessionePratica
808               WHERE IdPartecipante = OLD.IdPartecipante AND (Data -
809                     CURRENT_DATE) < 3)
810     THEN RAISE EXCEPTION 'Non puoi eliminare un partecipante che ha
811           effettuato un''adesione per una sessione pratica a meno di 3 giorni
812           prima della data';
813     END IF;
814
815     RETURN OLD;
816
817 END;
818
819 $$ LANGUAGE plpgsql;
820
821 CREATE TRIGGER trg_delete_partecipante_con_adesione
822 BEFORE DELETE ON Partecipante
823 FOR EACH ROW
824 EXECUTE FUNCTION fun_delete_partecipante_con_adesione();
825
826 -----
827
828 -- Restituisce TRUE se un corso      considerato "attivo":
829 -- 1 Ha almeno un iscritto
830 -- 2. La data odierna      tra DataInizio e la data dell'ultima sessione (o prima
831 -- della DataInizio)
832
833 CREATE OR REPLACE FUNCTION corso_attivo(id_corso_in INTEGER)

```

```

824 RETURNS BOOLEAN AS
825 $$
826 DECLARE
827     data_inizio_corso Corso.DataInizio%TYPE;
828     data_fine_corso Corso.DataInizio%TYPE;
829     num_iscritti INT;
830 BEGIN
831     -- Recupera la data di inizio del corso
832     SELECT DataInizio INTO data_inizio_corso FROM Corso WHERE IdCorso =
            id_corso_in;
833
834     -- Calcola la data dell'ultima sessione
835     SELECT MAX(Data) INTO data_fine_corso
836     FROM
837     (
838         SELECT Data FROM SessionePratica WHERE IdCorso = id_corso_in
839         UNION
840         SELECT Data FROM SessioneOnline WHERE IdCorso = id_corso_in
841     );
842
843     -- Conta gli iscritti al corso
844     SELECT COUNT(*)
845     INTO num_iscritti
846     FROM Iscrizioni
847     WHERE IdCorso = id_corso_in;
848
849     -- Verifica se il corso ha iscritti e se oggi tra inizio e fine corso
850     -- oppure prima dell'inizio
851     IF(num_iscritti > 0) THEN
852         IF (data_fine_corso >= CURRENT_DATE AND data_inizio_corso <=
            CURRENT_DATE) OR (CURRENT_DATE < data_inizio_corso)
853             THEN RETURN TRUE;
854         END IF;
855     ELSE
856         RETURN FALSE;
857     END IF;
858 END;
859 $$ LANGUAGE plpgsql;
860 -----
861
862 -- Blocca la eliminazione di uno chef con corsi attivi
863
864 CREATE OR REPLACE FUNCTION fun_delete_chef_con_corsi()
865 RETURNS TRIGGER AS
866 $$
867 BEGIN
868     -- Se esiste un corso attivo per quello chef
869     IF EXISTS (SELECT 1
870         FROM Corso
871         WHERE IdChef = OLD.IdChef AND corso_attivo(IdCorso))

```

```

872         -- Blocca cancellazione dello chef
873         THEN RAISE EXCEPTION 'Non puoi Eliminare uno Chef che ha corsi attivi.';
874     END IF;
875
876     RETURN OLD;
877 END;
878 $$ LANGUAGE plpgsql;
879
880 CREATE TRIGGER trg_delete_chef_con_corsi
881 BEFORE DELETE ON Chef
882 FOR EACH ROW
883 EXECUTE FUNCTION fun_delete_chef_con_corsi();
884
885 -----
886
887 -- Impedisce la modifica dei campi: IdCorso, Costo e IdChef
888
889 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_corso()
890 RETURNS TRIGGER AS
891 $$
892 BEGIN
893     RAISE EXCEPTION 'Non puoi modificare i campi IdCorso, Costo o IdChef.';
894 END;
895 $$ LANGUAGE plpgsql;
896
897 CREATE TRIGGER trg_blocca_aggiorna_corso
898 BEFORE UPDATE OF IdCorso, Costo, IdChef ON Corso
899 FOR EACH ROW
900 EXECUTE FUNCTION fun_blocca_aggiorna_corso();
901
902
903 -- Impedisce la modifica della DataInizio di un corso se sono gi presenti
904     sessioni collegate (pratiche o online)
905
906 CREATE OR REPLACE FUNCTION fun_blocca_aggiornamento_data_se_iniziato()
907 RETURNS TRIGGER AS
908 $$
909 BEGIN
910     -- Se esiste almeno una sessione associata al corso
911     IF EXISTS
912         (
913             SELECT 1 FROM SessionePratica WHERE IdCorso =
914                 NEW.IdCorso
915             UNION
916             SELECT 1 FROM SessioneOnline WHERE IdCorso =
917                 NEW.IdCorso
918         )
919     THEN
920         -- Blocca l'update della data di inizio
921         RAISE EXCEPTION 'Il corso e'' gi iniziato!! non puoi spostare
922             la data di inizio corso';

```



```

919         END IF;
920 END;
921 $$ LANGUAGE plpgsql;
922
923 CREATE TRIGGER trg_blocca_aggiornamento_data_se_iniziato
924 BEFORE UPDATE OF DataInizio ON Corso
925 FOR EACH ROW
926 EXECUTE FUNCTION fun_blocca_aggiornamento_data_se_iniziato();
927
928
929 -- Non possibile cancellare un corso attivo
930
931 CREATE OR REPLACE FUNCTION fun_blocca_delete_corso()
932 RETURNS TRIGGER AS
933 $$
934 BEGIN
935     IF corso_attivo(OLD.IdCorso) THEN
936         RAISE EXCEPTION 'Non puoi cancellare un corso attivo!!';
937     END IF;
938
939     RETURN OLD;
940 END;
941 $$ LANGUAGE plpgsql;
942
943 CREATE TRIGGER trg_blocca_delete_corso
944 BEFORE DELETE ON Corso
945 FOR EACH ROW
946 EXECUTE FUNCTION fun_blocca_delete_corso();
947
948 -----
949
950 -- Impedisce la modifica delle sessioni dei campi: IdSessione(pratica,online) e
951 IdCorso
952
953 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_sessioni()
954 RETURNS TRIGGER AS
955 $$
956 BEGIN
957     RAISE EXCEPTION 'Non puoi modificare i campi IdSessione e IdCorso.';
958 END;
959 $$ LANGUAGE plpgsql;
960
961 CREATE TRIGGER trg_blocca_aggiorna_sessioni_online
962 BEFORE UPDATE OF IdSessioneOnline, IdCorso ON SessioneOnline
963 FOR EACH ROW
964 EXECUTE FUNCTION fun_blocca_aggiorna_sessioni();
965
966 CREATE TRIGGER trg_blocca_aggiorna_sessioni_pratiche
967 BEFORE UPDATE OF IdSessionePratica, IdCorso ON SessionePratica
968 FOR EACH ROW
969 EXECUTE FUNCTION fun_blocca_aggiorna_sessioni();

```

```

969
970
971 -- Non    possibile cancellare una sessione gi    avvenuta se il corso    ancora
    attivo
972
973 CREATE OR REPLACE FUNCTION fun_blocca_delete_sessioni_passate()
974 RETURNS TRIGGER
975 AS
976 $$
977 BEGIN
978     -- Se il corso    attivo e la sessione    avvenuta
979     IF corso_attivo(OLD.IdCorso) AND OLD.Data <= CURRENT_DATE THEN
980         RAISE EXCEPTION 'Non puoi cancellare una sessione gi    avvenuta in un
            corso non ancora terminato';
981     END IF;
982
983     RETURN OLD;
984 END;
985 $$ LANGUAGE plpgsql;
986
987 CREATE TRIGGER trg_blocca_delete_sessioni_pratiche_passate
988 BEFORE DELETE ON SessionePratica
989 FOR EACH ROW
990 EXECUTE FUNCTION fun_blocca_delete_sessioni_passate();
991
992 CREATE TRIGGER trg_blocca_delete_sessioni_online_passate
993 BEFORE DELETE ON SessioneOnline
994 FOR EACH ROW
995 EXECUTE FUNCTION fun_blocca_delete_sessioni_passate();
996
997 -----
998
999 -- Non    possibile cancellare o modificare un argomento (    una tabella molti a
    molti globale e rappresenta keywords comuni per rappresentare le tematiche
    del corso)
1000
1001 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_argomento()
1002 RETURNS TRIGGER AS
1003 $$
1004 BEGIN
1005     RAISE EXCEPTION 'Non puoi modificare o cancellare un argomento!!!';
1006 END;
1007 $$ LANGUAGE plpgsql;
1008
1009 CREATE TRIGGER trg_blocca_aggiorna_argomento
1010 BEFORE DELETE OR UPDATE OF Nome ON Argomento
1011 FOR EACH ROW
1012 EXECUTE FUNCTION fun_blocca_aggiorna_argomento();
1013
1014 -----
1015

```

```

1016 -- Impedisce di modificare gli argomenti di un corso se      attivo o ha iscrizioni
1017 -- Una volta che un corso      attivo non      pi      possibile alterarne la lista di
      argomenti, per coerenza
1018
1019 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_argomenticorso()
1020 RETURNS TRIGGER AS
1021 $$
1022 BEGIN
1023     IF corso_attivo(OLD.IdCorso) THEN
1024         RAISE EXCEPTION 'Non puoi modificare i campi IdCorso, IdArgomento
            .';
1025     END IF;
1026 END;
1027 $$ LANGUAGE plpgsql;
1028
1029 CREATE TRIGGER trg_blocca_aggiorna_argomenticorso
1030 BEFORE UPDATE OF IdCorso, IdArgomento ON Argomenti_Corso
1031 FOR EACH ROW
1032 EXECUTE FUNCTION fun_blocca_aggiorna_argomenticorso();
1033
1034
1035 -- Impedisce la cancellazione d e l l ultimo argomento di un corso
1036
1037 CREATE OR REPLACE FUNCTION fun_blocca_cancellazione_argomento_corso()
1038 RETURNS TRIGGER AS
1039 $$
1040 DECLARE
1041     num_argomenti INT;
1042 BEGIN
1043     -- Conto gli argomenti del corso
1044     SELECT COUNT(*) INTO num_argomenti
1045     FROM Argomenti_Corso
1046     WHERE IdCorso = OLD.IdCorso;
1047
1048     -- Controllo il numero
1049     IF num_argomenti <= 1 THEN
1050         RAISE EXCEPTION 'Non puoi rimuovere l''ultimo argomento
            di un corso.';
1051     END IF;
1052
1053     RETURN OLD;
1054 END;
1055 $$ LANGUAGE plpgsql;
1056
1057 CREATE TRIGGER trg_blocca_cancellazione_argomento_corso
1058 BEFORE DELETE ON Argomenti_Corso
1059 FOR EACH ROW
1060 EXECUTE FUNCTION fun_blocca_cancellazione_argomento_corso();
1061
1062 -----
1063

```

```

1064 -- Impedisce la modifica di IdRicetta e IdChef nella tabella Ricetta
1065
1066 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_ricetta()
1067 RETURNS TRIGGER AS
1068 $$
1069 BEGIN
1070     RAISE EXCEPTION 'Non puoi modificare i campi IdRicetta e IdChef.';
1071 END;
1072 $$ LANGUAGE plpgsql;
1073
1074 CREATE TRIGGER trg_blocca_aggiorna_ricetta
1075 BEFORE UPDATE OF IdRicetta, IdChef ON Ricetta
1076 FOR EACH ROW
1077 EXECUTE FUNCTION fun_blocca_aggiorna_ricetta();
1078
1079
1080 -- Una ricetta non pu essere eliminata se utilizzata in una sessione pratica
    associata a un corso attivo.
1081 CREATE OR REPLACE FUNCTION fun_blocca_delete_ricetta_utilizzata()
1082 RETURNS TRIGGER AS
1083 $$
1084 BEGIN
1085     -- Se esiste una ricetta che in uso in un corso attivo...
1086     IF EXISTS (
1087
1088         SELECT 1
1089         FROM Preparazioni P JOIN SessionePratica SP ON P.
1090             IdSessionePratica = SP.IdSessionePratica
1091         WHERE P.IdRicetta = OLD.IdRicetta AND
1092             corso_attivo(SP.IdCorso)
1093     )
1094     -- Blocca la delete
1095     THEN RAISE EXCEPTION 'Non puoi cancellare una ricetta che in uso in
1096         una sessione pratica di un corso considerato attivo';
1097     END IF;
1098
1099     RETURN OLD;
1100 END;
1101 $$ LANGUAGE plpgsql;
1102
1103 CREATE TRIGGER trg_blocca_delete_ricetta_utilizzata
1104 BEFORE DELETE ON Ricetta
1105 FOR EACH ROW
1106 EXECUTE FUNCTION fun_blocca_delete_ricetta_utilizzata();
1107
1108 -----
1109
1110 -- Gli ingredienti rappresentano entit di base del sistema e non devono essere
    modificati
1111 -- o cancellati per evitare effetti a catena su ricette e preparazioni esistenti.
1112
1113 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_ingredient()

```

```

1110 RETURNS TRIGGER AS
1111 $$
1112 BEGIN
1113     RAISE EXCEPTION 'Non puoi modificare o cancellare un ingrediente!!!';
1114 END;
1115 $$ LANGUAGE plpgsql;
1116
1117 CREATE TRIGGER trg_blocca_aggiorna_ingrediente
1118 BEFORE DELETE OR UPDATE OF IdIngrediente, Nome, Origine ON Ingrediente
1119 FOR EACH ROW
1120 EXECUTE FUNCTION fun_blocca_aggiorna_ingrediente();
1121
1122 -----
1123
1124 -- Impedisce la modifica dei riferimenti nelle preparazioni pratiche
1125
1126 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_preparazioni()
1127 RETURNS TRIGGER AS
1128 $$
1129 BEGIN
1130     RAISE EXCEPTION 'Non puoi modificare i campi IdSessionePratica, IdRicetta
1131         .';
1132 END;
1133 $$ LANGUAGE plpgsql;
1134
1135 CREATE TRIGGER trg_blocca_aggiorna_preparazioni
1136 BEFORE UPDATE OF IdSessionePratica, IdRicetta ON Preparazioni
1137 FOR EACH ROW
1138 EXECUTE FUNCTION fun_blocca_aggiorna_preparazioni();
1139
1140 -- Impedisce l'aggiunta di preparazioni a sessioni gi avvenute per rispettare
1141     la cronologia degli eventi
1142
1143 CREATE OR REPLACE FUNCTION fun_preparazione_per_sessione_futura()
1144 RETURNS TRIGGER AS
1145 $$
1146 DECLARE
1147     Data_Session SessionePratica.Data%TYPE;
1148 BEGIN
1149     -- Prendo la data della sessione pratica
1150     SELECT Data INTO Data_Session
1151     FROM SessionePratica
1152     WHERE IdSessionePratica = NEW.IdSessionePratica;
1153
1154     -- Se passata blocco l'insert su preparazioni
1155     IF Data_Session <= CURRENT_DATE THEN
1156         RAISE EXCEPTION 'Non si pu aggiungere una ricetta ad una
1157             sessione gi avvenuta';
1158     END IF;
1159     RETURN NEW;

```

```

1158 END;
1159 $$ LANGUAGE plpgsql;
1160
1161 CREATE TRIGGER trg_preparazione_per_sessione_futura
1162 BEFORE INSERT ON Preparazioni
1163 FOR EACH ROW
1164 EXECUTE FUNCTION fun_preparazione_per_sessione_futura();
1165
1166 -----
1167
1168 -- Impedisce la modifica dei riferimenti nelle iscrizioni a un corso
1169
1170 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_iscrizioni()
1171 RETURNS TRIGGER AS
1172 $$
1173 BEGIN
1174     RAISE EXCEPTION 'Non puoi modificare i campi IdPartecipante, IdCorso.';
1175 END;
1176 $$ LANGUAGE plpgsql;
1177
1178 CREATE TRIGGER trg_blocca_aggiorna_iscrizioni
1179 BEFORE UPDATE OF IdPartecipante, IdCorso ON Iscrizioni
1180 FOR EACH ROW
1181 EXECUTE FUNCTION fun_blocca_aggiorna_iscrizioni();
1182
1183
1184 -- Impedisce la disiscrizione da un corso se si aderito a una sessione pratica
      imminente
1185
1186 CREATE OR REPLACE FUNCTION fun_impedisce_disiscrizione()
1187 RETURNS TRIGGER AS
1188 $$
1189 BEGIN
1190     -- Se esiste una adesione imminente da parte del partecipante a una
      sessione pratica blocca la disiscrizione
1191     IF EXISTS (SELECT 1
1192                FROM Adesioni A JOIN SessionePratica SP ON A.IdSessionePratica =
      SP.IdSessionePratica
1193                WHERE IdPartecipante = OLD.IdPartecipante AND (Data - CURRENT_DATE
      ) < 3)
1194     THEN RAISE EXCEPTION 'Non puoi disiscriverti dal corso perch hai aderito a
      una sessione che dista meno di 3 giorni';
1195     END IF;
1196     RETURN OLD;
1197 END;
1198 $$ LANGUAGE plpgsql;
1199
1200 CREATE TRIGGER trg_impedisce_disiscrizione
1201 BEFORE DELETE ON Iscrizioni
1202 FOR EACH ROW
1203 EXECUTE FUNCTION fun_impedisce_disiscrizione();

```

```

1204
1205
1206 -- Dopo una disiscrizione valida, elimina automaticamente adesioni future alle
      sessioni
1207
1208 CREATE OR REPLACE FUNCTION fun_gestisci_disiscrizione()
1209 RETURNS TRIGGER AS
1210 $$
1211 BEGIN
1212     -- Elimina tutte le adesioni del partecipante alle sessioni
      pratiche future del corso da cui si disiscritto
1213     -- Utilizza la clausola USING per fare riferimento a SessionePratica
      senza dover fare una join esplicita non permessa nella clausola WHERE
      di Adesioni,
1214     DELETE FROM Adesioni
1215     USING SessionePratica
1216     WHERE Adesioni.IdSessionePratica = SessionePratica.
      IdSessionePratica
1217     AND Adesioni.IdPartecipante = OLD.IdPartecipante
1218     AND SessionePratica.IdCorso = OLD.IdCorso
1219     AND SessionePratica.Data > CURRENT_DATE;
1220     RETURN OLD;
1221 END;
1222 $$ LANGUAGE plpgsql;
1223
1224 CREATE TRIGGER trg_gestisci_disiscrizione
1225 AFTER DELETE ON Iscrizioni
1226 FOR EACH ROW
1227 EXECUTE FUNCTION fun_gestisci_disiscrizione();
1228
1229
1230 --Non ci si pu iscrivere a un corso se gi iniziato o gi finito
1231
1232 CREATE OR REPLACE FUNCTION fun_iscrizione_corso_iniziato()
1233 RETURNS TRIGGER AS
1234 $$
1235 DECLARE
1236     Data_inizio_corso Corso.DataInizio%TYPE;
1237 BEGIN
1238     -- Prendo la data di inizio del corso associato all'iscrizione
1239     SELECT DataInizio INTO Data_inizio_corso
1240     FROM Corso
1241     WHERE IdCorso = NEW.IdCorso;
1242     -- Se gi iniziato o finito allora non blocco l'iscrizione
1243     IF Data_inizio_corso <= CURRENT_DATE THEN
1244         RAISE EXCEPTION 'Non ci si pu iscrivere ad un corso gi
      iniziato';
1245     END IF;
1246     RETURN NEW;
1247
1248 END;

```

```

1249 $$ LANGUAGE plpgsql;
1250
1251 CREATE TRIGGER trg_iscrizione_corso_iniziato
1252 BEFORE INSERT ON Iscrizioni
1253 FOR EACH ROW
1254 EXECUTE FUNCTION fun_iscrizione_corso_iniziato();
1255
1256 -- Se sono presenti sessioni pratiche allora non possibile cambiare il tipo
    del corso a non pratico tramite update
1257
1258 CREATE OR REPLACE FUNCTION fun_blocca_update_ispratico()
1259 RETURNS TRIGGER AS
1260 $$
1261 BEGIN
1262     -- Checka se stato impostato IsPratico a false nell'update e ci sono
        sessioni pratiche nel corso
1263     IF NEW.IsPratico = FALSE AND EXISTS (SELECT 1 FROM SessionePratica WHERE
        IdCorso = OLD.IdCorso) THEN
1264         RAISE EXCEPTION 'Non puoi rendere un corso non pratico se ci sono
            sessioni pratiche nel corso';
1265     END IF;
1266     RETURN NEW;
1267 END;
1268 $$ LANGUAGE plpgsql;
1269
1270 CREATE TRIGGER trg_blocca_update_ispratico
1271 BEFORE UPDATE OF IsPratico ON Corso
1272 FOR EACH ROW
1273 WHEN (OLD.IsPratico IS DISTINCT FROM NEW.IsPratico)
1274 EXECUTE FUNCTION fun_blocca_update_ispratico();
1275
1276 -----
1277
1278 -- Blocca l'aggiornamento dei campi IdPartecipante, IdSessionePratica e
    DataAdesione nella tabella Adesioni.
1279
1280 CREATE OR REPLACE FUNCTION fun_blocca_aggiorna_adesioni()
1281 RETURNS TRIGGER AS
1282 $$
1283 BEGIN
1284     RAISE EXCEPTION 'Non puoi modificare i campi IdPartecipante,
        IdSessionePratica, DataAdesione.';
1285 END;
1286 $$ LANGUAGE plpgsql;
1287
1288 CREATE TRIGGER trg_blocca_aggiorna_adesioni
1289 BEFORE UPDATE OF IdPartecipante, IdSessionePratica, DataAdesione ON Adesioni
1290 FOR EACH ROW
1291 EXECUTE FUNCTION fun_blocca_aggiorna_adesioni();
1292
1293

```



```

1294 -- Impedisce la cancellazione di un'adesione a una sessione pratica se mancano
      meno di 3 giorni dalla data della sessione.
1295 -- Garantisce che non si possano rimuovere adesioni all'ultimo momento,
      preservando l' organizzazione.
1296
1297 CREATE OR REPLACE FUNCTION fun_blocca_cancella_adesioni()
1298 RETURNS TRIGGER AS
1299 $$
1300 DECLARE
1301     data_sessione SessionePratica.Data%TYPE;
1302 BEGIN
1303     SELECT Data INTO data_sessione FROM SessionePratica WHERE
      IdSessionePratica = OLD.IdSessionePratica;
1304
1305     IF data_sessione - CURRENT_DATE <= 2 THEN
1306         RAISE EXCEPTION ' Non puoi cancellare una adesione a meno di 3
      giorni dalla sessione pratica!';
1307     END IF;
1308     RETURN OLD;
1309 END;
1310 $$ LANGUAGE plpgsql;
1311
1312 CREATE TRIGGER trg_blocca_cancella_adesioni
1313 BEFORE DELETE ON Adesioni
1314 FOR EACH ROW
1315 EXECUTE FUNCTION fun_blocca_cancella_adesioni();
1316
1317 -----

```