Basi di Dati e Laboratorio di Basi di Dati

Prof. A. Maratea
Anno Accademico 2022/2023

Progetto d'esame

Gestione di una catena di Palestre

Ippolito Gaetano

0124001867

Mabilia Francesco
0124001910

Vecchio Roberto
0124001871

Data di Consegna:
17/06/2023

Indice

Capitolo 1 – Analisi dei requisiti

1.1 – Sintesi dei requisiti

Si vuole realizzare una base di dati per rappresentare le informazioni relative ad una catena di palestre appartenenti all'azienda GYMBRO.

Essendo una catena di palestre, si vuole tenere traccia di tutte le sedi attive o non che hanno fatto parte della catena. Ogni sede sarà gestita da un responsabile, che si occuperà di assumere e licenziare il personale. Ogni responsabile viene assunto dal titolare.

Ogni dipendente è assegnato ad una sede specifica. Per la gestione dei dipendenti (Segretaria, Istruttore) si vuole tener traccia di tutte le presenze, indipendentemente che il dipendente sia un istruttore/segretaria, memorizzandone l'ora di ingresso, quella d'uscita e il turno svolto. Il responsabile di sede non dovrà registrare gli orari di accesso e di uscita.

Per ogni dipendente si vogliono registrare le generalità, le informazioni di contatto, la data di assunzione, il titolo di studio, la mansione, lo stipendio e la data di licenziamento, in quanto non si vogliono perdere i dati del dipendente che ha lavorato in sede.

Tutte le sedi prevedono diverse tipologie di iscrizioni alla sala attrezzi e ad altri corsi aggiuntivi. Verrà data la possibilità all'utente, di iscriversi a qualsiasi corso egli desidera a patto che non ci sia già una sottoscrizione che contenga quest'ultimo. Ogni tipologia di abbonamento può contenere uno o più corsi ed è categorizzata da una durata mensile e un costo.

Ogni sede è aperta tutti i giorni, tranne la domenica, dalle ore 7:00 alle ore 24:00.

Ogni sede oltre all'accesso alla sala attrezzi, eroga dei corsi, tenuti in diverse fasce orarie e distribuiti nel corso della settimana.

Ogni cliente allo scadere dell'anno solare dovrà esibire il certificato medico. Al momento della prima sottoscrizione, l'utente dovrà effettuare il pagamento di quest'ultima e dovrà esibire il certificato medico.

L'istruttore può compilare le schede relative ad un cliente, pertanto sarà dotato di un tablet che avrà lo scopo di mostrare la lista degli esercizi e di selezionarne 9, decidendone il numero di serie, di ripetizioni e se sono in super-serie. Ogni scheda ha traccia della data di inizio, della scadenza, dell'istruttore e dell'utente.

Ad ogni cliente verrà associata una tessera elettronica che avrà lo scopo di verificare la validità dell'abbonamento dell'utente, impedendone l'accesso alla palestra nel caso in cui quest'ultima fosse scaduta.

Ogni sede è composta da una o più sale che avranno lo scopo di ospitare i corsi. Inoltre, vi è la possibilità della presenza di piscine con una capienza massima propria.

L'utente, iscritto alla palestra, può effettuare, tramite la segretaria, una prenotazione, della durata di 1 ora, presso una piscina, purché non si sia già raggiunta la capienza massima prevista per l'orario indicato.

La segretaria, inoltre, ha la possibilità di vendere, fino a un massimo di 5 articoli con quantità variabile dei prodotti, fino ad esaurimento scorte. Per testimoniare l'avvenuta vendita dei prodotti, verrà rilasciato uno scontrino con i dati di quando sono stati acquistati i prodotti, in quale sede e il prezzo totale.

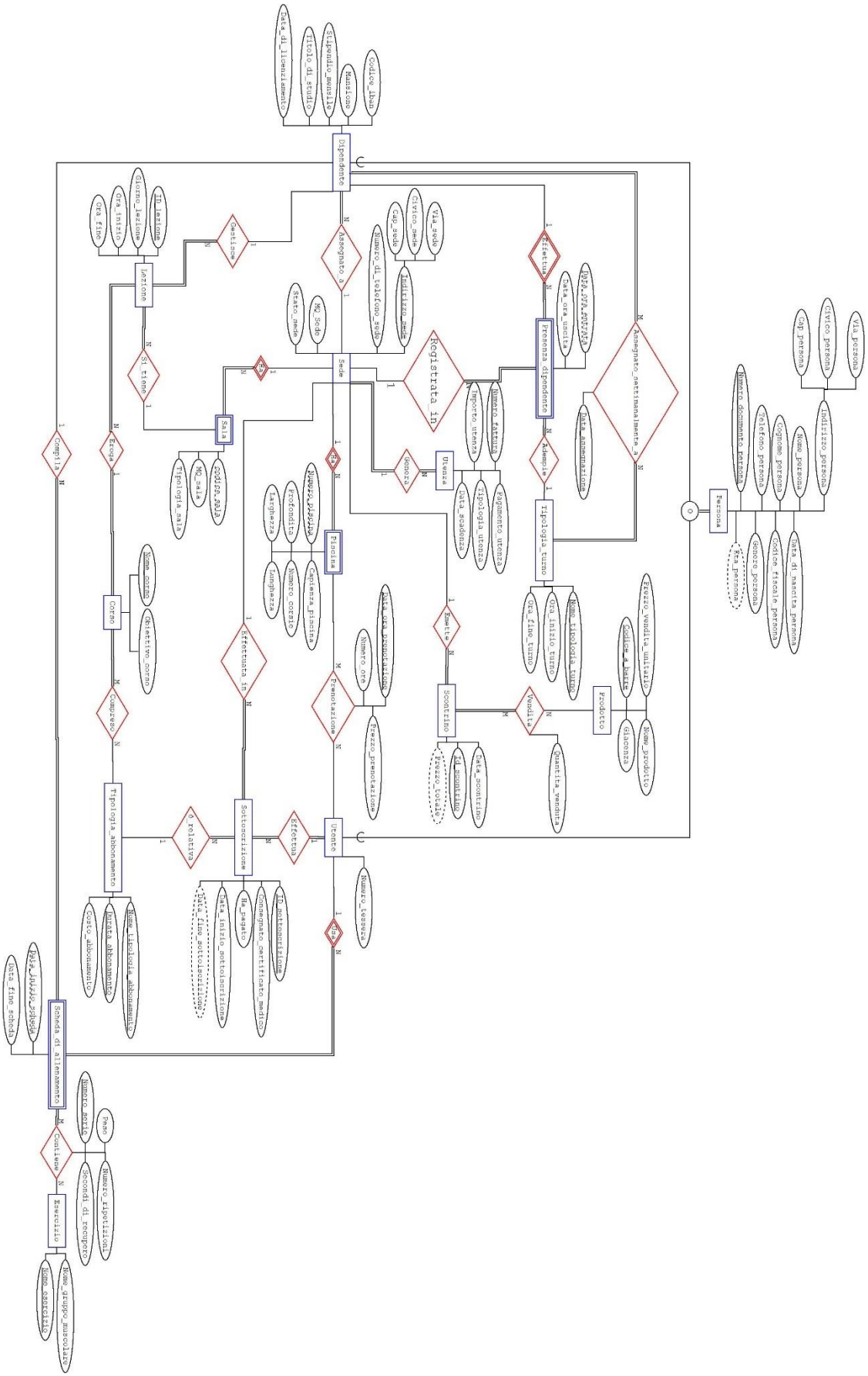
Si vuole, infine, tenere traccia delle informazioni delle utenze di ogni sede che vengono generate ogni 4 mesi.

1.2 – Glossario

TERMINI	DEFINIZIONE	SINONIMI	COLLEGAMENTI
Sede	Luogo dove è posta una palestra	Struttura	PISCINA, SALA, SCONTRINO, UTENZA, DIPENDENTE
Utente	Persona che si iscrive alla catena di sedi	Iscritto, cliente	PISCINA, SOTTOSCRIZIONE, SCHEDA DI ALLENAMENTO,
Dipendente	Lavoratore della catena di sedi	Segretaria, responsabile, istruttore	SCHEDA DI ALLENAMENTO, SEDE, TURNO SETTIMANALE, PRESENZA, CORSO
Prodotto	Articolo venduto all'interno della catena di palestre	Articolo	SEGRETARIA, UTENTE, SCONTRINO
Corso	Insieme di lezioni di una determinata disciplina		ISTRUTTORE, SALA, SOTTOSCRIZIONE, TIPOLOGIA ABBONAMENTO,
Titolare	Possessore della catena di palestre		RESPONSABILE, SEDE
Responsabile	Gestore di una sede	Responsabile di sede	SEDE
Abbonamento	Insieme di corsi	Tipologia abbonamento	UTENTE, SEGRETARIA, SOTTOSCRIZIONE, CORSO
Sottoscrizione	Sottoscrizione di partecipazione ad un abbonamento	Iscrizione	UTENTE, SEGRETARIA, SEDE, CORSO, ABBONAMENTO

Capitolo 2 – Diagramma EER e Relazionale

2.1 – Diagramma EER



2.1.1 – Analisi Entità ed attributi

ENTITÀ PERSONA

Questa entità rappresenta le persone che si iscrivono per le catene di sedi GYMBRO, oppure che lavorano per le sedi. Da questa breve descrizione, si può evincere che questa entità conterrà le informazioni della persona che può svolgere due ruoli fondamentali. Per questo motivo, essa sarà l'entità padre delle due entità figlie DIPENDENTE e UTENTE, in quanto entrambi sono persone ed entrambi hanno le stesse informazioni. Gli attributi di questa entità sono:

- *NUMERO_DOCUMENTO_PERSONA*: chiave primaria della persona; è numero del documento di identità;
- NOME_PERSONA: nome della persona;
- COGNOME_PERSONA: cognome della persona;
- CODICE_FISCALE_PERSONA: codice fiscale della persona;
- DATA_DI_NASCITA: data di nascita della persona;
- ~~ETA_PERSONA~~: età della persona (ottenibile dalla data di nascita);
- GENERE_PERSONA: genere biologico della persona
- INDIRIZZO_PERSONA: attributo multi-valore composto da:
    - VIA_PERSONA: via di residenza della persona;
    - CIVICO_PERSONA: civico di residenza della persona;
    - CAP_PERSONA: numero di CAP di residenza della persona;
- TELEFONO_PERSONA: numero di telefono della persona.

ENTITÀ DIPENDENTE { ENTITÀ FIGLIA DI PERSONA }

Questa entità rappresenta le informazioni dei dipendenti che lavorano per la catena di sedi GYMBRO. Gli attributi di questa entità sono:

- CODICE_IBAN: codice iban del dipendente;
- MANSIONE: ruolo del dipendente;
- STIPENDIO_MENSILE: stipendio del dipendente;
- TITOLO_DI_STUDIO: titolo di studio del dipendente;
- DATA_DI_LICENZIAMENTO: data di licenziamento se il dipendente viene licenziato.

ENTITÀ UTENTE { ENTITÀ FIGLIA DI PERSONA }

Questa entità rappresenta le informazioni degli utenti che sono iscritti alla catena di sedi GYMBRO. Gli attributi di questa entità sono:

- NUMERO_TESSERA: numero della tessera data al dipendente;

ENTITÀ SEDE

Questa entità rappresenta le informazioni delle sedi che appartengono alla catena di sedi GYMBRO e dove si svolgono tutte le attività. Gli attributi di questa entità sono:

- *INDIRIZZO_SEDE*: chiave primaria della sede, è l'indirizzo della sede. Questo attributo è un multi-valore:
  - *VIA_SEDE*: parte della chiave primaria, è la via della sede;
  - *CIVICO_SEDE:* parte della chiave primaria, è il civico della sede;
  - *CAP_SEDE:* parte della chiave primaria, è il numero cap della sede;
- MQ_SEDE: metri quadri della sede;
- NUMERO_DI_TELEFONO_SEDE: numero di telefono della sede;
- STATO_SEDE: stato della sede. Serve a identificare le sedi chiuse da quelle aperte.

ENTITÀ SALA { ENTITÀ DEBOLE DI SEDE }

Questa entità rappresenta le sale che sono contenute all'interno delle sedi. Per questo motivo, si può dedurre che non possono esistere sale se non esiste la sede e quindi questa entità è debole e identificata dalla sede. Gli attributi di questa entità sono:

- *CODICE SALA*: chiave debole della sala, è un codice per identificare la sala. La chiave completa di questa entità sarà {VIA_SEDE, CIVICO_SEDE, CAP_SEDE, CODICE_SALA};
- MQ_SALA: metri quadri della sala;
- TIPOLOGIA_SALA: tipologia di sala, ad esempio: "Attrezzi" per le sale attrezzi o "Corsi" per sale in cui si svolgono corsi.

ENTITÀ PISCINA { ENTITÀ DEBOLE DI SEDE }

Questa entità rappresenta le piscine che sono contenute all'interno delle sedi. Per questo motivo, si può dedurre che non possono esistere piscine se non esiste la sede e quindi questa entità è debole ed identificata dalla sede. Gli attributi di questa entità sono:

- *NUMERO PISCINA*: chiave debole della piscina, è un numero per identificare la piscina. La chiave completa di questa entità sarà {VIA_SEDE, CIVICO_SEDE, CAP_SEDE, NUMERO_PISCINA};
- LARGHEZZA: larghezza della piscina;
- LUNGHEZZA: lunghezza della piscina;
- PROFONDITÀ: profondità della piscina;
- CAPIENZA_PISCINA: capienza della piscina;
- NUMERO_CORSIE: numero di corsie della piscina.

ENTITÀ UTENZA

Questa entità rappresenta le utenze delle sedi, di cui si vuole tener traccia. Gli attributi di questa entità sono:

- *NUMERO_FATTURA*: chiave primaria dell'utenza, è un numero che identifica la utenza da pagare;
- IMPORTO_UTENZA: quant'è l'importo dell'utenza da pagare;
- PAGAMENTO_UTENZA: se il pagamento è stato effettuato oppure no;
- TIPOLOGIA_UTENZA: che tipo di utenza bisogna pagare;
- DATA_SCADENZA: la data della scadenza dell'utenza.

ENTITÀ SCONTRINO

Questa entità rappresenta gli scontrini emessi dalla catena di sedi GYMBRO. Gli attributi di questa entità sono:

- *ID_SCONTRINO*: chiave primaria dello scontrino, è un numero sequenziale dei scontrini generati;
- DATA_SCONTRINO: data in cui lo scontrino viene emesso;
- ~~PREZZO_TOTALE~~: prezzo totale dei prodotti acquistati e registrati nello scontrino. Valore ottenuto dal prezzo del prodotto e dalla quantità venduta del prodotto.

ENTITÀ PRODOTTO

Questa entità rappresenta i prodotti che la catena di sedi GYMBRO vende all'interno delle sue strutture. Gli attributi di questa entità sono:

- *CODICE_A_BARRE*: chiave primaria del prodotto, è codice per identificare un prodotto;
- PREZZO_DI_VENDITA_UNITARIO: prezzo di vendita del singolo prodotto;
- NOME_PRODOTTO: nome del prodotto;
- GIACENZA: scorte del prodotto.

ENTITÀ LEZIONE

Questa entità rappresenta le lezioni che si tengono all'interno delle sale delle sedi GYMBRO. Gli attributi di questa entità sono:

- *ID_LEZIONE*: chiave primaria della lezione, è un numero di sequenza che aumenta con le lezioni;
- GIORNO_LEZIONE: giorno in cui si terrà la lezione;
- ORA_INIZIO: ora in cui si terrà la lezione;
- ORA_FINE: ora in cui finirà la lezione.

ENTITÀ CORSO

Questa entità rappresenta i corsi che la catena di sedi GYMBRO offre ai propri utenti. Gli attributi di questa entità sono:

- *NOME_CORSO*: chiave primaria del corso, è il nome del corso;
- OBIETTIVO_CORSO: è l'obiettivo del corso.

ENTITÀ TIPOLOGIA_ABBONAMENTO

Questa entità rappresenta gli abbonamenti che la catena di sedi GYMBRO offre ai propri utenti. Gli attributi di questa entità sono:

- *NOME_TIPOLOGIA_ABBONAMENTO*: chiave primaria della tipologia abbonamento, è il nome della tipologia abbonamento;
- *DURATA_ABBONAMENTO*: chiave primaria della tipologia abbonamento, è la durata dell'abbonamento in termini di mesi;
- COSTO_ABBONAMENTO: costo dell'abbonamento.

ENTITÀ SOTTOSCRIZIONE

Questa entità rappresenta le sottoscrizioni che sono effettuate dagli utenti e relative ad una sede di appartenenza ed una tipologia di abbonamento. La sottoscrizione in una sede non implica che l'utente possa recarsi solo nella suddetta, infatti, la presenza della chiave della sede in questa entità, ha il puro scopo di specificare dov'è avvenuta la sottoscrizione. Gli attributi di questa entità sono:

- *ID_SOTTOSCRIZIONE*: chiave primaria della sottoscrizione, è un numero sequenziale che aumenta con le iscrizioni;
- DATA_INIZIO_SOTTOSCRIZIONE: data di quando l'utente si è sottoscritto;
- ~~DATA_FINE_SOTTOSCRIZIONE~~: data di fine sottoscrizione (ottenibile dalla data inizio + la durata dell'abbonamento);
- HA_PAGATO: specifica se l'utente ha pagato oppure non ha pagato la sottoscrizione;
- CERTIFICATO_MEDICO: specifica se l'utente ha consegnato il certificato medico oppure non l'ha consegnato.

ENTITÀ SCHEDA DI ALLENAMENTO { ENTITÀ DEBOLE DI UTENTE }

Questa entità rappresenta le schede di allenamento degli utenti. Visto che la scheda non viene generata se non esiste un utente, essa sarà un'entità debole. Gli attributi di questa entità sono:

- *DATA_INIZIO_SCHEDA*: chiave debole della scheda, è una data per identificare l'inizio della scheda di allenamento. La chiave completa di questa entità sarà {NUMERO_DOCUMENTO_PERSONA(UTENTE), DATA_INIZIO_SCHEDA};
- DATA_FINE_SCHEDA: data per la fine della scheda.

ENTITÀ ESERCIZIO

Questa entità rappresenta gli esercizi. Gli attributi di questa entità sono:

- *NOME_ESERCIZIO*: chiave primaria dell'esercizio, è il nome dell'esercizio;
- NOME_GRUPPO_MUSCOLARE: è la zona di interesse che viene sforzata dall'esercizio.

ENTITÀ PRESENZA_DIPENDENTE { ENTITÀ DEBOLE DI DIPENDENTE }

Questa entità rappresenta le presenze effettuate dai dipendenti. Una presenza può esistere solo se un dipendente la effettua, per cui si può dire con certezza che la presenza è un'entità debole. Gli attributi di questa entità sono:

- *DATA_ORA_ENTRATA*: chiave debole di presenza dipendente, è la data e l'ora della presenza del dipendente. La chiave completa di questa entità sarà {NUMERO_DOCUMENTO_PERSONA(DIPENDENTE), DATA_ORA_ENTRATA};
- DATA_ORA_USCITA: è la data e ora dell'uscita del dipendente.

ENTITÀ TIPOLOGIA_TURNO

Questa entità rappresenta gli esercizi. Gli attributi di questa entità sono:

- *NOME_TIPOLOGIA_TURNO*: chiave primaria di tipologia turno, è il nome della tipologia turno;
- ORA_INIZIO: è l'orario in cui inizia un turno;
- ORA_FINE: è l'orario in cui finisce un turno.

2.1.2 – Analisi Associazioni

In questo paragrafo, verranno mostrate tutte le associazioni presenti nel diagramma EE/R della catena di sedi GYMBRO. Verranno elencati in rosso tutte le associazioni con molteplicità M:N.

ASSEGNATO_SETTIMANALMENTE_A

- Molteplicità: M:N;
- Totalità: entrambi i lati, in quanto tutte le tipologie turno sono assegnate a uno o più dipendenti, ogni dipendente viene assegnato ad una o più tipologie turno;
- Nel relazionale viene cambiato il nome con "assegnazione_settimanale".
- Chiave: [data_assegnazione, numero_documento_persona(Dipendente), nome_tipologia_turno]

PRENOTAZIONE

- Molteplicità: M:N
- Totalità: totalità da nessuna delle due parti, in quanto non è detto che un utente deve prenotare una piscina e non è detto che tutte le piscina vengano prenotate da almeno un utente;
- Chiave: [numero_documento_persona(Utente), numero_piscina, via_sede, civico_sede, cap_sede, data_ora_prenotazione].

COMPRESO

- Molteplicità: M:N
- Totalità: totalità da parte di corso, in quanto tutti i corsi sono inclusi in almeno una tipologia_abbonamento;
- Chiave: [nome_corso, nome_tipologia_abbonamento, durata_abbonamento].

VENDITA

- Molteplicità: M:N
- Totalità: totalità da parte dello scontrino, in quanto ogni scontrino contiene almeno un prodotto venduto, ma un prodotto non è detto che sia venduto;
- Chiave: [id_scontrino, codice_a_barre].

CONTIENE

- Molteplicità: M:N
- Totalità: totalità da parte della scheda di allenamento, in quanto ogni scheda di allenamento contiene degli esercizi, ma non tutti gli esercizi sono contenuti in una scheda;
- Chiave: [numero_documento_persona(utente), data_inizio_scheda].

EMETTE

- Molteplicità: 1:N
- Totalità: totalità da parte dello scontrino, dato che uno scontrino viene emesso per forza da una sede ma non è detto che tutte le sedi emettono uno scontrino.

GENERA

- Molteplicità: 1:N
- Totalità: totalità da entrambe le parti, tutte le sedi generano utenze ed almeno un'utenza è generata in una sede.

ASSEGNATO_A

- Molteplicità: 1:N
- Totalità: totalità da parte del dipendente, dato che tutti i dipendenti sono assegnati ad una sede, ma non è detto che ci sia una sede con dipendenti, per esempio una sede appena creata.

GESTISCE

- Molteplicità: 1:N
- Totalità: totalità da parte della lezione, dato che tutte le lezioni sono gestite dai dipendenti, ma non tutti i dipendenti gestiscono una lezione, come per esempio i responsabili o le segretarie.

SI_TIENE

- Molteplicità: 1:N
- Totalità: totalità da parte di lezione, dato che tutte le lezioni si tengono in una sala, ma non tutte le sale tengono una lezione, come le sale attrezzi.

HA { SEDE -> PISCINA }

- Molteplicità: 1:N
- Totalità: totalità da parte di piscina, dato che ogni piscina si trova in una Sede ma non tutte le Sedi hanno una piscina.
- Questa associazione si riferisce all'associazione tra "SEDE" e "PISCINA"
- Questa associazione è identificante nei confronti dell'entità debole PISCINA.

HA { SEDE -> SALA }

- Molteplicità: 1:N
- Totalità: totalità da parte di sala, dato che ogni sala si trova in una Sede ma non tutte le Sedi hanno una sala.
- Questa associazione si riferisce all'associazione tra "SEDE" e "PISCINA"
- Questa associazione è identificante nei confronti dell'entità debole SALA.

EROGA

- Molteplicità: 1:N
- Totalità: totalità da entrambi i lati, in quanto la lezione è sempre erogata da un corso ed il corso eroga delle lezioni.

EFFETTUATA_IN

- Molteplicità: 1:N
- Totalità: totalità da parte di sottoscrizione, in quanto tutte le sottoscrizioni sono effettuate in una sede ma non tute le sede effettuano sottoscrizioni.

EFFETTUA

- Molteplicità: 1:N
- Totalità: totalità da entrambi i lati, in quanto tutti gli utenti effettuano almeno una sottoscrizione e tutte le sottoscrizioni sono effettuate da utenti.

È_RELATIVA

- Molteplicità: 1:N
- Totalità: totalità da parte di sottoscrizione, in quanto tutte le sottoscrizioni sono relative a delle tipologie abbonamento, ma non tutte le tipologie abbonamento solo relative a delle sottoscrizioni, come, ad esempio, le tipologie abbonamento a cui nessuno si sottoscrive.

COMPILA

- Molteplicità: 1:N
- Totalità: totalità da parte della scheda di allenamento, in quanto tutte le schede di allenamento sono compilate da Dipendenti, ovvero gli istruttori, ma non tutti i dipendenti compilano delle schede di allenamento. s

USA

- Molteplicità: 1:N
- Totalità: totalità da parte di scheda di allenamento, in quanto tutte le schede di allenamento generate sono usate da utenti, ma non tutti gli utenti usano schede di allenamento;
- Questa associazione è identificante nei confronti dell'entità debole SCHEDA_DI_ALLENAMENTO.

EFFETTUA

- Molteplicità: 1:N
- Totalità: totalità da parte della presenza dipendente, in quanto tutte le presenze del dipendente sono effettuate da dipendenti, ma non tutti i dipendenti effettuano presenze, come, per esempio, i dipendenti che si assentano o i responsabili;
- Questa associazione è identificante nei confronti dell'entità debole PRESENZA_DIPENDENTE.

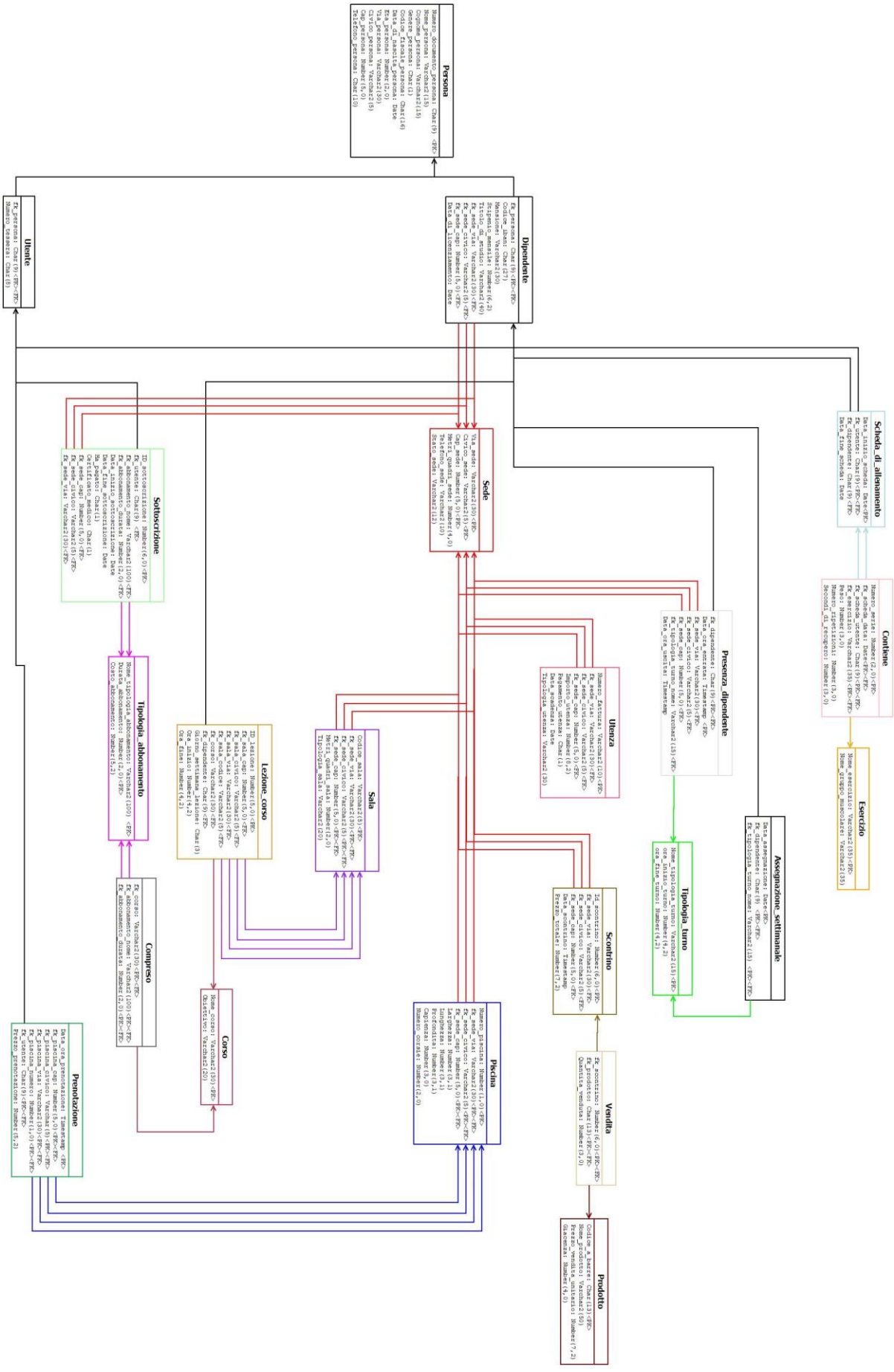
ADEMPIE

- Molteplicità: 1:N
- Totalità: totalità da parte di presenza dipendente, in quanto tutte le presenze del dipendente sono adempiute da almeno una tipologia turno, ma non tutte le tipologie turno adempiono ad una presenza, come, per esempio, le tipologie turno che non vengono assegnate a nessuno.

REGISTRATA_IN

- Molteplicità: 1:N
- Totalità: totalità da parte di presenza dipendente, in quanto tutte le presenze del dipendente sono registrate in una sede, ma non tutte le sede registrano delle presenze, siccome potrebbero esistere delle sedi nello stato "NON ATTIVO".

2.2 – Diagramma Relazionale



In questo capitolo si mostrerà il passaggio dalla modellazione concettuale a quella logica, passando, quindi dal diagramma EE/R a quello Relazionale. Lo schema relazionale comprende più relazioni, quindi è formato da:

- R_i - Relazioni: l'insieme di tutte le associazioni, per cui verranno raffigurati le tabelle di tutte le entità;
- V_j - Integrity Condition: l'insieme dei vincoli di integrità, ovvero i vincoli che garantiscono l'integrità del database;
- D_k - Functional Dipendence: l'insieme delle dipendenze funzionali;

Successivamente mostriamo lo schema relazionale, ottenuto dalla traduzione dello schema EE/R, seguendo le regole per la corretta traduzione.

2.2.1 – Analisi Tabelle

Nel diagramma relazione sono presenti:

- 11 Entità:
  - PERSONA;
  - SEDE;
  - UTENZA;
  - SCONTRINO;
  - PRODOTTO;
  - LEZIONE_CORSO;
  - SOTTOSCRIZIONE;
  - TIPOLOGIA_ABBONAMENTO;
  - CORSO, ESERCIZIO;
  - TIPOLOGIA_TURNO.
- 2 entità figlie:
  - UTENTE;
  - DIPENDENTE.
- 4 entità deboli:
  - SALA;
  - PISCINA;
  - SCHEDA_DI_ALLENAMENTO;
  - PRESENZA_DIPENDENTE.
- 5 associazioni M:N:
  - CONTIENE;
  - ASSEGNAZIONE_SETTIMANALE;
  - PRENOTAZIONE;
  - COMPRESO;
  - VENDITA.

Per un totale di 22 Tabelle.

Si è voluto specificare il tipo "Timestamp" nelle tabelle PRESENZA_DIPENDENTE e PRENOTAZIONE in quanto si voleva sottolineare l'importanza di avere come informazione l'orario, visto che si svolgono alcune operazioni fondamentali con esso. Per questo motivo, i campi che hanno come nome "data" utilizzeranno il tipo "Date"; mentre i campi che hanno come nome "data_ora" utilizzeranno il tipo "Timestamp".

Capitolo 3 – Utenti e loro categorie

Si è ipotizzato che per il nostro Database vi fossero 5 figure importanti, ovvero coloro i quali interagiscono maggiormente con il sistema: Titolare, Responsabile di sede, Segretaria, Istruttore e l'iscritto.

Per evitare problemi di omonimia con la tavola degli utenti, l'Utente, che è stato citato dai paragrafi precedenti, prenderà il nome di Iscritto in questo e nel prossimo paragrafo.

Il Titolare sarà colui che avrà ogni singolo permesso all'interno del sistema, essendo lui il proprietario della catena di sedi GYMBRO, per cui sarà l'amministratore.

Il Responsabile di sede è un utente che ha lo scopo di supervisionare la propria sede e di essere presente all'interno di essa nel caso si verifichi un incidente. Avrà sicuramente ruoli gestionali della propria sede, ottenendo tutte quelle operazioni di gestione della struttura, come: gestione dei corsi, dei turni dei dipendenti, assunzioni e licenziamenti e altro.

La segretaria è importante per la compilazione di quelle richieste che l'iscritto effettua, per esempio l'iscrizione o la prenotazione o il salvataggio dei dati di una vendita di un prodotto all'interno di una sede.

L'istruttore ha un ruolo più semplice rispetto a quelli elencati finora, in quanto avrà l'unisco scopo di dirigere un corso o di compilare una scheda.

Infine, l'iscritto il cui unico metodo di interazione con il sistema è tramite la sua tessera della palestra. Presupponendo che gli venga data al momento dell'iscrizione, essa sarà fondamentale per permettergli l'accesso all'interno della sede. Quindi non interagisce come i dipendenti, ma ha comunque un'operazione che agisce all'interno del sistema e che fa partire nel caso in cui voglia entrare all'interno di una delle sedi GYMBRO. Tramite questa operazione gli verrà notificato che tutte le sue sottoscrizioni sono scadute e che per tal motivo il suo accesso è stato negato.

3.1 – Categoria utenti

A fronte dell'analisi effettuata nel paragrafo precedente, verrà riportata la tavola degli utenti contenente il tipo, i volumi ed i permessi degli utenti sopra elencati:

Utente	Tipo	Volume	Permessi
Titolare	Amministratore	1	ALL
Responsabile	Comune	15	RWU
Istruttore	Comune	45	RW
Segretaria	Comune	30	RWU
Iscritto	Comune	N	-

Bisogna considerare che, per politiche aziendali, vi sono un massimo di: 1 Responsabile, 3 istruttori, 2 segretarie ed 1 titolare. Tenendo conto che al momento della creazione del database vi sono 15 sedi, questi volumi sono ottenuti dalla moltiplicazione tra il numero degli utenti per il numero delle sedi.

3.2 – Operazioni degli utenti

Una volta analizzati quali sono gli utenti che interagiscono con il sistema e le loro possibile azioni, si potrà procedere con il Diagramma dei casi d'uso, il quale mostra graficamente ciò che è stato descritto finora:

LEGENDA:

Significa che le operazioni sono semplici e non richiedono l'utilizzo di più tabelle;

Significa che le operazioni sono di scrittura e richiamano 3 o più tabelle.

Per rendere la visibilità del diagramma migliore, si è voluto collegare il titolare ai suoi casi d'uso, evitando di collegarlo a tutti i casi d'uso presenti.

Di seguito verranno elencate le tavole descrittori delle operazioni e le tavole delle operazioni degli utenti, principalmente dei casi d'uso segnati in rosso:

- In parentesi verrà indicato il nome utilizzato per la procedura.

Operazione	Effettua assunzione (Assunzione)
Scopo	Inserimento di una Persona come Dipendente
Argomenti	Numero_documento_persona, Nome_persona, Cognome_persona, Genere_persona, Codice_fiscale, Data_di_nascita_persona, Via_persona, Civico_persona, Cap_persona, telefono_persona, Codice_iban, Mansione, Stipendio_mensile, Titolo_di_studio, Via_sede, Civico_sede, Cap_sede
Risultato	Inserimento dei dati / Errore
Errori	Sede non attiva
Usa	PERSONA, DIPENDENTE, SEDE
Modifica	Persona, Dipendente
Prima	Non è inserita la persona e il dipendente
Poi	È inserita la persona e il dipendente

Operazione	Creazione tipologia abbonamento (creazione_abbonamento)
Scopo	Inserimento di una nuova tipologia abbonamento e di un nuovo corso se esso non esistesse
Argomenti	nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento, nome_corso1, obiettivo1 [… nome_corso4, obiettivo4]
Risultato	Inserimento dati in Tipologia_abbonamento, Corso e Compreso / Errore
Errori	-
Usa	COMPRESO, TIPOLOGIA_ABBONAMENTO, CORSO
Modifica	Compreso, Tipologia_abbonamento, Corso
Prima	Non sono inseriti dati della tipologia abbonamento, non esiste il corso (se nuovo), non sono inseriti dati in Compreso
Poi	Sono inseriti i dati della tipologia abbonamento, inserito il corso (se nuovo), inseriti dati in Compreso

Operazione	Crea/Compila una scheda_di_allenamento (compila_scheda)
Scopo	Creazione di una scheda di allenamento e compilazione della tabella Contiene
Argomenti	data_inizio_scheda, data_fine_scheda, fk_scheda_utente, fk_scheda_dipendente, numero_serie, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero [… numero_serie9, fk_esercizio9, peso9, numero_ripetizioni9, secondi_di_recupero9]
Risultato	Inseriti dati in Scheda_di_allenamento e in Contiene / non inseriti dati in Scheda_di_allenamento e Contiene
Errori	Mansione del dipendente diversa da “Istruttore”, Utente non iscritto al corso di “Sala”
Usa	CONTIENE, COMPRESO, DIPENDENTE, SOTTOSCRIZIONE, SCHEDA_DI_ALLENAMENTO
Modifica	Scheda_di_allenamento, Contiene
Prima	Non ci sono dati inseriti in Scheda_di_allenamento e Contiene
Poi	Dati inseriti in Scheda_di_allenamento e Contiene

Operazione	Registra turno (registra_turno)
Scopo	Registrare la presenza del dipendente in entrata e in uscita, tenendo conto della sua assegnazione settimanale
Argomenti	numero_dipendente
Risultato	Inserimento della presenza in entrata o in uscita / non inserimento della presenza
Errori	Dipendente che risulta licenziato
Usa	DIPENDENTE, ASSEGNAZIONE_SETTIMANALE, TIPOLOGIA_TURNO, PRESENZA_DIPENDENTE, SEDE
Modifica	Presenza_dipendente
Prima	Dati della presenza non inseriti
Poi	Dati della presenza inseriti o in entrata o in uscita

Operazione	Sottoscrizione utente (sottoscrizione_utente)
Scopo	Registrare i dati di una persona come Utente (se non si è mai iscritta) e a quale abbonamento si sottoscrive
Argomenti	Numero_persona, nome_abbonamento, via_sede, civico_sede, cap_sede, codice_fiscale, data_di_nascita, nome, cognome, genere, via_persona, civico_persona, cap_persona, telefono_persona
Risultato	Inserimento dei dati in Persona e Utente / non inserimento dei dati in Persona e Utente
Errori	L'utente è già sottoscritto alla tipologia abbonamento fornita, la sede è nello stato 'NON ATTIVO'
Usa	SEDE, PERSONA, UTENTE, SOTTOSCRIZIONE
Modifica	Persona, Utente, Sottoscrizione
Prima	Dati di persona, utente e sottoscrizione dell'utente non esistenti
Poi	Dati di persona, utente e sottoscrizione esistenti

Operazione	Vende prodotto (vendita_prodotti)
Scopo	Registrare i dati di una vendita e dello scontrino rilasciato
Argomenti	Sede_via, sede_civico, sede_cap, codice_a_barre1, quantita_venduta1 [… codice_a_barre5, quantita_venduta5]
Risultato	Inserimento dello scontrino e della vendita effettuata / non inserimento dello scontrino e della vendita
Errori	Sede nello stato 'NON ATTIVO'
Usa	SCONTRINO, PRODOTTO, VENDITA, SEDE
Modifica	Scontrino, vendita
Prima	Dati dello scontrino e della vendita non esistenti
Poi	Dati dello scontrino e della vendita esistenti

Di seguito la tavola dei volumi delle operazioni:

Operazione	Tipo	Volume	Periodo
Effettua assunzione	**B**	120	MESE
Creazione tipologia abbonamento	**B**	30	ANNO
Crea/Compila una scheda_di_allenamento	**B**	25	SETTIMANA
Registra turno	**B**	90	GIORNO
Sottoscrizione utente	**B**	35	SETTIMANA
Vende prodotto	**B**	25	SETTIMANA

Il volume è ottenuto dal numero di volte in cui l'operazione è stata lanciata, contando anche le volte in cui non hanno avuto successo.

3.3 – Volumi

In questo paragrafo, verrà riportata la tabella dei volumi, in cui si mostra il numero vero simile di tuple presenti in ciascuna tabella una volta che il Database sia a regime. Verranno riportati anche i dati degli ipotetici incrementi in un periodo di tempo prefissato.

Bisogna specificare che vi è una politica di aggiornamento sulla tabella Prenotazione, in cui le tuple vecchie di un anno, a partire dalla data odierna, vengono cancellate ogni mese. Per cui non vi sarà un effettivo incremento.

Tutti questi calcoli sono stati ottenuti considerando sempre il caso migliore di ogni Tabella.

Di seguito la tabella:

Tabella	Tipo	Volume	Incrementa	Periodo
Persona	E	22680	1137	Anno
Dipendente	E(Figlia)	120	12	Anno
Utente	E(Figlia)	22500	1125	Anno
Sede	E	20	1	Anno
Sala	ED	60	3	Anno
Piscina	ED	40	2	Anno
Scheda_di_allenamento	ED	18000	900	Anno
Contiene	A	162000	8100	Mensile
Esercizio	E	150	5	Anno
Presenza_dipendente	ED	40000	100	Giorno
Tipologia_turno	E	10	5	Anno
Assegnazione_settimanale	A	100	100	Settimana
Sottoscrizione	E	22500	1125	Anno
Tipologia_abbonamento	E	30	2	Anno
Compreso	A	90	6	Anno
Corso	E	15	1	Anno
Lezione_corso	E	600	600	Settimana
Prenotazione	A	1000	-	Mese
Utenza	E	180	9	Anno
Scontrino	E	30	2	Anno
Vendita	A	90	6	Anno
Prodotto	E	15	1	Anno

I volumi sono stati ottenuti in questo modo:

- PERSONA: si è ottenuto il volume di 22680 calcolando prima il volume dei dipendenti e degli utenti, in quanto sono persone. Se considerassimo 20 sedi, in cui vi sono 6 dipendenti al suo interno, avremmo un totale di 120 dipendenti. Gli utenti sono ottenuti considerano che vi sono 75 utenti per corso (al momento 15 corsi) distribuiti nelle 20 sedi, ipotizzando che tutti i corsi vengano occupati, ottenendo: (75 * 15) * 20 = 22500. Se a questo si aggiungono i dipendenti otteniamo 22680 persone;
- SALA: si è ottenuto il volume di 60 ipotizzando che per ogni sede che viene aperta vi saranno un massimo di 3 sale. Se stiamo considerano almeno 20 sedi, allora 20 * 3 = 60;
- PISCINA: si è ottenuto il volume di 40 ipotizzando che per ogni sede che viene aperta vi saranno un massimo di 2 piscine. Se stiamo considerano almeno 20 sedi, allora 20 * 2 = 40;
- SCHEDA DI ALLENAMENTO: si è ottenuto il volume 18000 considerando che almeno 80% degli utenti richiedono la scheda o sono iscritti a sala. Per cui (22500 * 80) / 100 = 18000;
- CONTIENE: si è ottenuto 162000 ipotizzando che ogni scheda di allenamento ha almeno 9 esercizi. Per cui: 18000 * 9 = 162000;
- ESERCIZIO: si è ottenuto 150 in quanto non si crede ad un effettivo aumento del numero degli esercizi e ad un futuro cambiamento. Si è ritenuto che una volta che verranno inseriti un quantitativo alto di esercizi, esse potrebbero non aumentare quasi mai. L'incremento vuole sottolineare un caso raro in cui verrà inserito un nuovo esercizio;
- PRESENZA DIPENDENTE: si è ottenuto 40000 ipotizzando che 100 dipendenti (responsabile esclusi) si presentino nelle proprie sedi durante i loro turni. Avendo 20 sedi e considerando che ogni dipendente effettua almeno 5 presenze a settimana (20 mensili), avremo 100 * 20 * (4 * 5) = 40000;
- SOTTOSCRIZIONE: si è ottenuto 22500 in quanto un utente per essere tale deve essere sottoscritto;
- TIPOLOGIA ABBONAMENTO: si è ottenuto 30 in quanto per ogni corso vi è l'abbonamento al corso singolo. Ipotizzando che vi sono 15 corsi abbiamo 15 tipologie abbonamento certe. A questo sono stati aggiunti altri 15 nel caso in cui vengano create delle tipologie abbonamento che comprendono quel corso singolo più altri corsi;
- COMPRESO: si è ottenuto 90 con il seguente ragionamento: se per 15 corsi si hanno certi 15 tipologie abbonamento, è sicuro che avremo 15 compresi. Se verrà aggiunta una tipologia abbonamento che non aumenta il numero dei corsi e che ne comprende 2 esistenti, allora avremo 16 tipologie - 15 corsi – 17 compresi, in quanto la tipologia comprende due corsi. Quindi, se consideriamo 30 tipologie, abbiamo 30 compresi + il numero dei possibili corsi che comprendono. Se decidiamo che ipoteticamente ogni tipologia comprende 3 corsi, allora 30 * 3 = 90;
- LEZIONE CORSO: si è ottenuto 600 in quanto si sono considerati 5 lezioni al giorno per 20 sedi per 6 giorni. Per cui: 5 * 6 * 20 = 600;
- UTENZA: si è ottenuto 180 considerando che vengono generate 3 utenze 3 volte l'anno per le 20 sedi, avremo: 3 * 3 * 20
- SCONTRINO - VENDITA - PRODOTTO: si è seguita la stessa logica di Tipologia abbonamento, compreso e corso.

Capitolo 4 – Vincoli d'integrità

4.1 – Vincoli statici

Nel seguente paragrafo verranno elencati tutti i vincoli d'integrità statici individuati all'intero del sistema. I vincoli d'integrità statici sono vincoli che impediscono a degli attributi di assumere valori che potrebbero corrompere l'integrità del database. Questi vincoli sono indipendenti dal tempo, quindi si attivano solo ad un inserimento errato all'interno di una tabella. Di seguito, verranno riportati tutti i vincoli statici, con condizioni particolari, per cui verranno omessi gli attributi con solo vincolo NOT NULL e tutte le chiavi primarie e esterne:

- PERSONA
  - numero_documento_persona: questo attributo, che rappresenta la chiave primaria, dovrà essere necessariamente di 9 caratteri;
  - genere_persona: questo attributo dovrà essere di massimo 1 carattere e solo 'M' o 'F';
  - codice_fiscale_persona: questo attributo dovrà essere UNIQUE per ogni persona e di massimo 16 caratteri;
  - data_di_nascita: al fine di poter calcolare l'età, esso dovrà essere NOT NULL;
  - telefono_persona: questo attributo è di 10 caratteri;
- SEDE
  - metri_quadri_sede: questo attributo dovrà essere maggiore di 0;
  - stato_sede: questo attributo potrà avere solo due valori 'ATTIVO' o 'NON ATTIVO';
- TIPOLOGIA_ABBONAMENTO
  - durata_abbonamento: essendo la durata espressa in numero di mesi, essa dovrà essere assumere i valori che vanno da 1 a 12;
  - costo_abbonamento: non può esistere un costo di un abbonamento che sia uguale a zero;
- DIPENDENTE
  - codice_iban: ogni dipendente dovrà avere il proprio codice iban;
  - stipendio_mensile: è permesso uno stipendio del dipendente che va da 800 a 5000 euro;
- UTENTE
  - numero_tessera: questo attributo dovrà essere necessariamente di 8 caratteri.
- UTENZA
  - importo_utenza: questo attributo potrà assumere solo valori maggiori di 0;
  - pagamento_utenza: questo attributo è formato da un solo carattere che dovrà essere necessariamente 'S' o 'N'.
- SCONTRINO
  - prezzo_totale: non potrà esserci un valore minore o uguale di 0 per questo campo.
- PISCINA
  - larghezza: la lunghezza deve essere necessariamente maggiore di 0;
  - lunghezza: la lunghezza deve essere necessariamente maggiore di 0;
  - profondità: la profondità deve essere necessariamente maggiore di 0;
  - capienza: la capienza deve essere necessariamente maggiore di 0;
  - numero_corsie: vi è un minimo i 2 corsie ed un massimo di 10 per le piscine.
- SALA
  - metri_quadri_sala: così come sede, anche sala avrà la stessa regola, ovvero che i metri quadri dovranno essere maggiori di 0.
- LEZIONE_CORSO
  - giorno_settimana_lezione: questo attributo potrà avere solo i valori 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT', in quanto rappresenterà i giorni della settimana. Escludiamo 'SUN' perché di domenica non si potranno avere lezioni in quanto la sede è chiusa.

- SOTTOSCRIZIONE
    - ha_pagato: questo attributo rappresenta l'avvenuto pagamento oppure la possibile scadenza della sottoscrizione. Pertanto, dovrà avere i valori 'S' o 'N';
    - certificato_medico: questo attributo rappresenta l'avvenuta consegna del certificato medico oppure la possibile scadenza del certificato. Pertanto, dovrà avere i valori 'S' o 'N';
- CONTIENE
    - peso: questo attributo potrà avere valori maggiori o uguali a 0;
    - numero_di_ripetizioni: questo attributo potrà avere valori maggiori o uguali a 0;
    - secondi_di_recupero: questo attributo potrà avere valori maggiori o uguali a 0.
- VENDITA
    - quantita_venduta: non si potranno vendere 0 prodotti, quindi il valore deve essere maggiore di 0.
- PRENOTAZIONE
    - prezzo_prenotazione: non si può avere una prenotazione con un costo inferiore o uguale a 0.

4.2 – Vincoli dinamici

I vincoli d'integrità dinamici, invece, riguardano gli attributi che variano in funzione del tempo o che vengono combinati, con dati di natura temporale. Di seguito, sono elencati, i soli vincoli dinamici gestiti tramite triggers; non vengono riportati in quanto ovvi quelli che gestiscono gli auto-incrementi delle chiavi artificiali:

- Età persona: possono essere registrate solo persone tra i 6 e i 100 anni;
- Massime ore lavorative settimanali: un dipendente può lavorare per un massimo di 48h settimanali;
- Sovrapposizioni dei turni: un dipendente non può essere assegnato ad una nuova tipologia turno le cui ore lavorative comprendono il turno già assegnatogli per quel giorno;
- Orari di prenotazione piscina: le prenotazioni per la piscina hanno durata fissa di 1h, in cui non verrà data la possibilità all'utente di prenotarsi ulteriormente nella fascia oraria in cui ha già effettuato una prenotazione;
- Capienza massima piscina: ogni piscina ha la propria capienza massima. Questa servirà, per motivi di sicurezza, ad impedire che vengano generate più prenotazioni della capienza massima della piscina. Ovviamente, questo vincolo vale per quella fascia oraria in cui si ha un alto numero di prenotazioni;
- Numero lezioni giornaliere: possono esserci un massimo di 5 lezioni giornaliere in una sede;
- Massimo dipendenti per sede: possono esserci al massimo 6 dipendenti per sede (1 responsabile, 3 istruttori, 2 segretarie);
- Età dipendente: un dipendente per essere definito tale deve avere almeno 18 anni compiuti;
- Sottoscrizione: un utente può sottoscriversi una sola volta ad un corso e quindi non può sottoscrivere più di una volta diverse tipologie abbonamento che contengono lo stesso corso;
- Prodotti in giacenza: una qualsiasi persona non potrà acquistare più prodotti di quanti ne sono contenuti in giacenza;
- Massime sottoscrizioni corso: un corso non potrà essere sottoscritto da più di 75 utenti.

Capitolo 5 – Normalizzazione

Le forme normali permettono di valutare la qualità di uno schema nonché la ridondanza che può generare.

1. Uno schema è detto in prima forma normale se tutti i suoi attributi sono atomici, quindi se non presenta attributi multi-valore, composti o loro combinazioni. In altre parole, essa richiede che il dominio di un attributo comprenda solo valori semplici, indivisibili e che il valore di qualsiasi attributo, sia un valore singolo nel suo dominio. Il rispetto di questi requisiti è implicito dal momento che si tratta di uno schema flat-relational model. La conformità alla prima forma normale è banalmente garantita dall'assenza di attributi non atomici o multivalore;
2. La seconda forma normale riguarda le chiavi multi-attributo: tutti gli attributi non primi dipendono in maniera completa dalla chiave. Cioè, tutti gli attributi che non fanno parte della chiave e che non sono univoci (non UNIQUE), dipendono funzionalmente, da tutti quelli che compongono la chiave, ovvero non ci sono dipendenze parziali con la suddetta. Si basa, quindi, sul concetto di dipendenza funzionale completa. Una dipendenza funzionale X da Y, in uno schema R, è una dipendenza funzionale completa se la rimozione di qualsiasi attributo $A \in X$ da X, comporta che la dipendenza non sussista più. La conformità alla seconda forma normale è garantita dalla dipendenza completa di ogni attributo non primo da ogni chiave composta e dalla conformità alla prima forma normale;
3. La terza forma normale richiede che tutte le dipendenze funzionali siano determinate da chiavi oppure su attributi primi. Essa non garantisce la conservazione delle dipendenze, ma solo quella dei dati, infatti è sempre raggiungibile. Si basa sul concetto di dipendenza funzionale transitiva. Uno schema è in terza forma normale quando soddisfa la seconda forma normale e quando nessun attributo non-primo dipende in modo transitivo dalla chiave primaria. La conformità alla terza forma normale è garantita dalla totale indipendenza di ogni attributo non primo da altri attributi non primi e dalla conformità alla seconda forma normale;
4. La conformità alla forma normale di BCNF è garantita dalla conformità alla terza forma normale al quale si aggiunge la restrizione che gli attributi debbano dipendere funzionalmente solo dalla chiave.

Capitolo 6 – Possibili estensioni

Le possibili estensioni del sistema sono:

- Implementazione dei fornitori per i prodotti;
- Permettere all'utente di reperire la propria scheda in formato digitale;
- Possibilità di aggiungere più esercizi ad una scheda già creata;
- Possibilità modificare degli esercizi aggiunti all'interno di una scheda già creata;
- Salvataggio dei dati riguardanti la collaborazione con enti esterni (es. ditta di pulizie);

Capitolo 7 – Implementazione

L'ambiente di sviluppo usato è Oracle 11g Express Edition.

È consigliabile effettuare i seguenti passi:

Creare l'utente Amministratore con tutti i privilegi e connettersi;

1. Creare le tabelle e le sequenze per le chiavi artificiali;
2. Creare i triggers per l'auto-incremento delle precedenti sequenze create;
3. Popolare le tabelle (ordine numerato);
4. Inserire triggers, procedure, views e schedulers;
5. Creare gli utenti con i relativi privilegi di ruolo.

7.1 – Data Definition Language

```
-----CREAZIONE TABELLE-----

-----PERSONA-----
CREATE TABLE Persona (
    -- ATTRIBUTI --
    numero_documento_persona CHAR(9) PRIMARY KEY,
    nome_persona          VARCHAR2(15),
    cognome_persona       VARCHAR2(15),
    genere_persona        CHAR(1) CHECK ( UPPER(genere_persona) = 'M' OR UPPER(genere_persona) = 'F' ),
   codice_fiscale_persona  CHAR(16) UNIQUE NOT NULL,
   data_di_nascita        DATE NOT NULL,
   eta_persona            NUMBER(2, 0),
   via_persona            VARCHAR2(30),
   civico_persona         VARCHAR2(5),
   cap_persona            NUMBER(5, 0),
   telefono_persona       CHAR(10)
);

-----SEDE-----
CREATE TABLE Sede (
   -- ATTRIBUTI –
   via_sede        VARCHAR2(30),
   civico_sede     VARCHAR2(5),
   cap_sede        NUMBER(5, 0),
   metri_quadri_sede NUMBER(4, 0) CHECK ( metri_quadri_sede > 0 ),
   telefono_sede    VARCHAR2(10),
   stato_sede      VARCHAR(12) DEFAULT 'ATTIVO' CHECK ( UPPER(stato_sede) = 'ATTIVO'
                                                    OR UPPER(stato_sede) = 'NON ATTIVO' ),


   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_sede PRIMARY KEY ( via_sede, civico_sede, cap_sede )
);

-----ESERCIZIO-----
CREATE TABLE Esercizio (
   -- ATTRIBUTI --
   nome_esercizio      VARCHAR2(35) PRIMARY KEY,
   nome_gruppo_muscolare VARCHAR2(35)
);
```

```
-----TIPOLOGIA TURNO-----
CREATE TABLE Tipologia_turno (
   -- ATTRIBUTI --
   nome_tipologia_turno   VARCHAR2(15) PRIMARY KEY,
   ora_inizio_turno    NUMBER(4,2) NOT NULL,
   ora_fine_turno      NUMBER(4,2) NOT NULL

);

-----PRODOTTO-----
CREATE TABLE Prodotto (
   -- ATTRIBUTI --
   codice_a_barre         CHAR(13) PRIMARY KEY,
   nome_prodotto          VARCHAR2(50),
   prezzo_vendita_unitario NUMBER(7, 2) NOT NULL,
   giacenza               NUMBER(4, 0) NOT NULL
);

-----CORSO-----
CREATE TABLE Corso(
   -- ATTRIBUTI --
   nome_corso VARCHAR2(30) PRIMARY KEY,
   obiettivo  VARCHAR2(20)
);

-----TIPOLOGIA ABBONAMENTO-----
CREATE TABLE Tipologia_abbonamento (
   -- ATTRIBUTI --
   nome_tipologia_abbonamento VARCHAR2(100),
   durata_abbonamento         NUMBER(2, 0) CHECK ( durata_abbonamento BETWEEN 1 AND 12 ),
   costo_abbonamento          NUMBER(5, 2) CHECK ( costo_abbonamento > 0 ) NOT NULL,

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_tipologia_abbonamento PRIMARY KEY ( nome_tipologia_abbonamento, durata_abbonamento )
);

-----DIPENDENTE-----
CREATE TABLE Dipendente (
   -- ATTRIBUTI –
   fk_persona          CHAR(9) PRIMARY KEY,
   codice_iban         CHAR(27) NOT NULL UNIQUE,
   mansione            VARCHAR(30) NOT NULL,
   stipendio_mensile   NUMBER(6, 2) DEFAULT 1000.00 CHECK (stipendio_mensile BETWEEN 800.00 AND 5000.00),
   titolo_di_studio    VARCHAR2(40),
   data_di_licenziamento DATE DEFAULT NULL,
   fk_sede_via         VARCHAR2(30),
   fk_sede_civico      VARCHAR2(5),
   fk_sede_cap         NUMBER(5, 0),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_persona_dipendente FOREIGN KEY ( fk_persona )
      REFERENCES Persona ( numero_documento_persona ) ON DELETE CASCADE,
   CONSTRAINT fk_sede_dipendente FOREIGN KEY ( fk_sede_via, fk_sede_civico, fk_sede_cap )
      REFERENCES Sede ( via_sede, civico_sede, cap_sede )
);
```

```
-----UTENTE-----
CREATE TABLE Utente (
   -- ATTRIBUTI --
   fk_persona     CHAR(9) PRIMARY KEY,
   numero_tessera CHAR(8) UNIQUE,

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_utente_persona FOREIGN KEY ( fk_persona )
      REFERENCES Persona ( numero_documento_persona ) ON DELETE CASCADE
);

-----UTENZA-----
CREATE TABLE Utenza (
   -- ATTRIBUTI --
   numero_fattura   VARCHAR2(10) PRIMARY KEY,
   fk_sede_via      VARCHAR2(30),
   fk_sede_civico   VARCHAR2(5),
   fk_sede_cap      NUMBER(5, 0),
   importo_utenza   NUMBER(8, 2) CHECK ( importo_utenza > 0 ) NOT NULL,
   pagamento_utenza CHAR(1) DEFAULT 'N' CHECK ( UPPER(pagamento_utenza) = 'S'
                                                OR UPPER(pagamento_utenza) = 'N' ),
   data_scadenza    DATE NOT NULL,
   tipologia_utenza VARCHAR2(30),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_sede_utenza FOREIGN KEY ( fk_sede_via, fk_sede_civico, fk_sede_cap )

      REFERENCES Sede ( via_sede, civico_sede, cap_sede )
);

-----PRESENZA DIPENDENTE-----
CREATE TABLE Presenza_dipendente (
   -- ATTRIBUTI --
   data_ora_entrata         TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   data_ora_uscita          TIMESTAMP,
   fk_dipendente            CHAR(9),
   fk_sede_via              VARCHAR2(30) NOT NULL,
   fk_sede_civico           VARCHAR2(5) NOT NULL,
   fk_sede_cap              NUMBER(5, 0) NOT NULL,
   fk_tipologia_turno_nome  VARCHAR2(15) NOT NULL,

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_presenza_dipendente PRIMARY KEY (data_ora_entrata, fk_dipendente ),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_presenza_dipendente FOREIGN KEY ( fk_dipendente )
      REFERENCES Dipendente ( fk_persona ), -- on delete default --
   CONSTRAINT fk_sede_presenza FOREIGN KEY ( fk_sede_via, fk_sede_civico, fk_sede_cap )
      REFERENCES Sede ( via_sede, civico_sede, cap_sede ),  -- on delete default --
   CONSTRAINT fk_tipologia_turno_presenza FOREIGN KEY ( fk_tipologia_turno_nome)
      REFERENCES Tipologia_turno ( nome_tipologia_turno ) -- on delete default --
);
```

```
-----SCHEDA DI ALLENAMENTO-----
CREATE TABLE Scheda_di_allenamento (
   -- ATTRIBUTI --
   data_inizio_scheda DATE,
   fk_utente        CHAR(9),
   fk_dipendente     CHAR(9) NOT NULL,
   data_fine_scheda   DATE,

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_scheda_di_allenamento PRIMARY KEY ( data_inizio_scheda, fk_utente ),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_dipendente_scheda FOREIGN KEY ( fk_dipendente )
      REFERENCES Dipendente ( fk_persona ), -- on delete default --
   CONSTRAINT fk_utente_scheda FOREIGN KEY ( fk_utente )
      REFERENCES Utente ( fk_persona ) -- on delete default --
);

-----SCONTRINO-----
CREATE TABLE Scontrino(
   -- ATTRIBUTI --
   id_scontrino NUMBER(6,0) PRIMARY KEY,
   fk_sede_via VARCHAR2(30) NOT NULL,
   fk_sede_civico VARCHAR2(5) NOT NULL,
   fk_sede_cap NUMBER(5,0) NOT NULL,
   data_scontrino TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   prezzo_totale NUMBER(7,2) CHECK (prezzo_totale > 0),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_sede_scontrino FOREIGN KEY(fk_sede_via, fk_sede_civico, fk_sede_cap)
            REFERENCES Sede(via_sede, civico_sede, cap_sede) -- on delete default --
);

-----PISCINA-----
CREATE TABLE Piscina (
   -- ATTRIBUTI --
   numero_piscina NUMBER(1, 0),
   fk_sede_via    VARCHAR2(30),
   fk_sede_civico VARCHAR2(5),
   fk_sede_cap    NUMBER(5, 0),
   larghezza      NUMBER(3, 1) DEFAULT 25.0 CHECK ( larghezza > 0 ),
   lunghezza      NUMBER(3, 1) DEFAULT 50.0 CHECK ( lunghezza > 0 ),
   profondita     NUMBER(3, 1) DEFAULT 2.0 CHECK ( profondita > 0 ),
   capienza       NUMBER(3, 0) CHECK ( capienza > 0 ) NOT NULL,
   numero_corsie  NUMBER(2, 0) DEFAULT 5 CHECK ( numero_corsie BETWEEN 2 AND 10 ),

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_piscina PRIMARY KEY ( numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap ),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_sede_piscina FOREIGN KEY ( fk_sede_via, fk_sede_civico, fk_sede_cap )
      REFERENCES Sede ( via_sede, civico_sede, cap_sede ) -- on delete default --
);
```

```
-----SALA-----
CREATE TABLE Sala (
   -- ATTRIBUTI --
   codice_sala      VARCHAR2(5),
   fk_sede_via      VARCHAR2(30),
   fk_sede_civico   VARCHAR2(5),
   fk_sede_cap      NUMBER(5, 0),
   metri_quadri_sala  NUMBER(2, 0) CHECK ( metri_quadri_sala > 0 ),
   tipologia_sala   VARCHAR2(20) NOT NULL,

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_sala PRIMARY KEY ( codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap ),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_sede_sala FOREIGN KEY ( fk_sede_via, fk_sede_civico, fk_sede_cap )
      REFERENCES Sede ( via_sede,   civico_sede, cap_sede ) -- on delete default --
);

-----LEZIONE CORSO-----
CREATE TABLE lezione_corso (
   -- ATTRIBUTI --
   id_lezione            NUMBER(5, 0) PRIMARY KEY,
   fk_sala_codice        VARCHAR2(5) NOT NULL,
   fk_sala_via           VARCHAR2(30) NOT NULL,
   fk_sala_civico        VARCHAR2(5) NOT NULL,
   fk_sala_cap           NUMBER(5, 0) NOT NULL,
   fk_corso              VARCHAR2(30) NOT NULL,
   fk_dipendente         CHAR(9) NOT NULL,
   giorno_settimana_lezione CHAR(3) CHECK ( upper(giorno_settimana_lezione)
                                            IN ( 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT' ) ) NOT NULL,
   ora_inizio            NUMBER(4, 2) NOT NULL,
   ora_fine              NUMBER(4, 2) NOT NULL,

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_sala_lezione FOREIGN KEY ( fk_sala_via, fk_sala_civico, fk_sala_cap, fk_sala_codice )
      REFERENCES sala ( fk_sede_via, fk_sede_civico, fk_sede_cap, codice_sala ) ON DELETE CASCADE,
   CONSTRAINT fk_corso_lezione FOREIGN KEY ( fk_corso )
      REFERENCES corso ( nome_corso ) ON DELETE CASCADE,
   CONSTRAINT fk_istruttore_lezione FOREIGN KEY ( fk_dipendente )
      REFERENCES dipendente ( fk_persona ) -- on delete default --
);

-----SOTTOSCRIZIONE-----
CREATE TABLE Sottoscrizione (
   -- ATTRIBUTI --
   id_sottoscrizione        NUMBER(6, 0) PRIMARY KEY,
   fk_utente                CHAR(9) NOT NULL,
   fk_abbonamento_nome      VARCHAR2(100) NOT NULL,
   fk_abbonamento_durata    NUMBER(2, 0) NOT NULL,
   fk_sede_via              VARCHAR2(30) NOT NULL,
   fk_sede_civico           VARCHAR2(5) NOT NULL,
   fk_sede_cap              NUMBER(5, 0) NOT NULL,
   data_inizio_sottoscrizione DATE DEFAULT current_date NOT NULL,
   data_fine_sottoscrizione   DATE,
   ha_pagato                CHAR(1) DEFAULT 'S' CHECK ( UPPER(ha_pagato) = 'S' OR UPPER(ha_pagato) = 'N' ),
   certificato_medico       CHAR(1) DEFAULT 'S' CHECK ( UPPER(certificato_medico) = 'S'
                                                        OR UPPER(certificato_medico) = 'N' ),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_utente_sottoscrizione FOREIGN KEY ( fk_utente )
      REFERENCES Utente ( fk_persona ),  -- on delete default --
```

```
  CONSTRAINT fk_sede_sottoscrizione FOREIGN KEY ( fk_sede_via, fk_sede_civico, fk_sede_cap )
    REFERENCES Sede ( via_sede, civico_sede, cap_sede ), -- on delete default --
  CONSTRAINT fk_abbonamento_sottoscrizione FOREIGN KEY ( fk_abbonamento_nome, fk_abbonamento_durata )
    REFERENCES Tipologia_abbonamento ( nome_tipologia_abbonamento, durata_abbonamento )
);

-----ASSEGNAZIONE SETTIMANALE-----
CREATE TABLE Assegnazione_settimanale (
  -- ATTRIBUTI --
  data_assegnazione       DATE,
  fk_dipendente           CHAR(9),
  fk_tipologia_turno_nome  VARCHAR2(15),

  -- VINCOLI DI CHIAVE PRIMARIA --
  CONSTRAINT pk_assegnazione_settimanale PRIMARY KEY(data_assegnazione, fk_dipendente,
                                                     fk_tipologia_turno_nome),

  -- VINCOLI DI CHIAVE ESTERNA --
  CONSTRAINT fk_dipendente_assegnazione FOREIGN KEY ( fk_dipendente )
    REFERENCES Dipendente ( fk_persona ), -- on delete default --
  CONSTRAINT fk_turno_assegnazione FOREIGN KEY ( fk_tipologia_turno_nome )
    REFERENCES Tipologia_turno ( nome_tipologia_turno )
);

-----CONTIENE-----
CREATE TABLE Contiene (
  -- ATTRIBUTI --
  numero_serie       NUMBER(2, 0),
  fk_scheda_data     DATE,
  fk_scheda_utente   CHAR(9),
  fk_esercizio       VARCHAR2(35),
  peso               NUMBER(3, 0) CHECK ( peso >= 0 ),
  numero_ripetizioni  NUMBER(3, 0) CHECK ( numero_ripetizioni >= 0 ),
  secondi_di_recupero NUMBER(3, 0) CHECK ( secondi_di_recupero >= 0 ) NOT NULL,

  -- VINCOLI DI CHIAVE PRIMARIA --
  CONSTRAINT pk_contiene PRIMARY KEY ( numero_serie, fk_scheda_data,  fk_scheda_utente,   fk_esercizio ),

  -- VINCOLI DI CHIAVE ESTERNA --
  CONSTRAINT fk_scheda_contiene FOREIGN KEY ( fk_scheda_data, fk_scheda_utente )
    REFERENCES Scheda_di_allenamento ( data_inizio_scheda, fk_utente ) ON DELETE CASCADE,
  CONSTRAINT fk_esercizio_contiene FOREIGN KEY ( fk_esercizio )
    REFERENCES Esercizio ( nome_esercizio ) ON DELETE CASCADE
);

-----VENDITA-----
CREATE TABLE Vendita (
  -- ATTRIBUTI --
  fk_scontrino    NUMBER(6, 0),
  fk_prodotto     CHAR(13),
  quantita_venduta NUMBER(3, 0) CHECK ( quantita_venduta > 0) NOT NULL,

  -- VINCOLI DI CHIAVE PRIMARIA --
  CONSTRAINT pk_vendita PRIMARY KEY ( fk_scontrino,  fk_prodotto ),

  -- VINCOLI DI CHIAVE ESTERNA --
  CONSTRAINT fk_scontrino_vendita FOREIGN KEY ( fk_scontrino )
    REFERENCES Scontrino ( id_scontrino ), -- on delete default --
  CONSTRAINT fk_prodotto_vendita FOREIGN KEY ( fk_prodotto )
    REFERENCES Prodotto ( codice_a_barre )
);
```

```
-----PRENOTAZIONE-----
CREATE TABLE Prenotazione (
   -- ATTRIBUTI --
   data_ora_prenotazione   TIMESTAMP,
   fk_utente          CHAR(9),
   fk_piscina_numero   NUMBER(1,0),
   fk_piscina_via      VARCHAR2(30),
   fk_piscina_civico   VARCHAR2(5),
   fk_piscina_cap      NUMBER(5, 0),
   prezzo_prenotazione NUMBER(5, 2) CHECK ( prezzo_prenotazione > 0 ) NOT NULL,

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_prenotazione PRIMARY KEY ( data_ora_prenotazione, fk_utente, fk_piscina_numero,
                              fk_piscina_via, fk_piscina_civico, fk_piscina_cap ),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_piscina_prenotazione FOREIGN KEY ( fk_piscina_numero, fk_piscina_via, fk_piscina_civico,
                                                    fk_piscina_cap )
      REFERENCES Piscina ( numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap ) ON DELETE CASCADE,
   CONSTRAINT fk_utente_prenotazione FOREIGN KEY ( fk_utente )
      REFERENCES Utente ( fk_persona ) -- on delete default --
);

-----COMPRESO-----
CREATE TABLE Compreso(
   -- ATTRIBUTI --
   fk_corso VARCHAR2(30),
   fk_abbonamento_nome VARCHAR2(100),
   fk_abbonamento_durata NUMBER(2,0),

   -- VINCOLI DI CHIAVE PRIMARIA --
   CONSTRAINT pk_compreso PRIMARY KEY(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata),

   -- VINCOLI DI CHIAVE ESTERNA --
   CONSTRAINT fk_corso_compreso FOREIGN KEY(fk_corso) REFERENCES Corso(nome_corso) ON DELETE CASCADE,
   CONSTRAINT fk_tipologia_abbonamento_compreso FOREIGN KEY(fk_abbonamento_nome,
                                                            fk_abbonamento_durata)
      REFERENCES Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento) ON DELETE CASCADE
);
```

7.1.1 - Sequenze

```
-- Creazione sequenze per chiavi artificiali --

-- SEQUENZA PER "Sottoscrizione.id_sottoscrizione" --
CREATE SEQUENCE id_sottoscrizione_seq
START WITH 1
INCREMENT BY 1;

-- SEQUENZA PER "Lezione_corso.id_lezione" --
CREATE SEQUENCE id_lezione_seq
START WITH 1
INCREMENT BY 1;

-- SEQUENZA PER "Scontrino.id_scontrino" --
CREATE SEQUENCE id_scontrino_seq
START WITH 1
INCREMENT BY 1;

-- SEQUENZA PER "Utente.numero_tessera" --
CREATE SEQUENCE numero_tessera_seq
START WITH 1
INCREMENT BY 1;
```

7.2 – Data Manipulation Language

Di seguito, vengono mostrate solo alcuni inserimenti a titolo dimostrativo (comunque visionabile nei file .sql nella cartella Popolamento):

```
-- Popolamento sede --
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via salvator rosa', '150', 80136, 150, '0815485478');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Mar Rosso', '5', 24123, 150, '0815485468');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Giuseppe Verdi', '1', 80121, 140, '0813454713');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Piazza Garibaldi', '33', 80136, 110, '0816434373');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Viale Manzoni', '53', 80121, 155, '0815413578');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Corso Europa', '17', 80016, 150, '0815565423');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Napoli', '3', 24126, 98, '0815975008');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Dante', '7', 61121, 90, '0810005478');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Viale delle industrie', '22', 80142, 170, '0815320478');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Cesare', '38', 44126, 79, '0815497023');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Rosa', '4', 24122, 34, '0810052153');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Verdi', '23', 24126, 98, '0815834097');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Via Massimo', '76', 61126, 90, '0810067315');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Viale Vittoria', '1', 41126, 170, '0893420478');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Piazza di Spagna', 'A3', 29121, 79, '0856842023');
INSERT INTO Sede(via_sede, civico_sede, cap_sede, metri_quadri_sede, telefono_sede) VALUES('Piazza Verdi', '50', 29132, 34, '0810974023');

-- Popolamento Persona –
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CA78525DV', 'Paolo', 'Lucci', 'M', 'PLALCC97B14F839E', TO_DATE('14-02-1997', 'DD-MM-YYYY'), 'Via salvator rosa', '189', 80016, '3405698752');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CA78534DK', 'Luisa', 'Pinto', 'F', 'LSUPTN97G18F839X', TO_DATE('18-08-1997', 'DD-MM-YYYY'), 'Via unione sovietica', '101', 80135, '3407854152');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CA78548DL', 'Alberico', 'Sasso', 'M', 'ALBSSS02G18F839L', TO_DATE('18-08-2002', 'DD-MM-YYYY'), 'Via giacinto gigante', '200', 80051, '3409874152');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CA79834DZ', 'Gaetano', 'Rossi', 'M', 'GTNRSS98M19F839X', TO_DATE('19-07-1998', 'DD-MM-YYYY'), 'Corso vittorio emanuele', '500', 80062, '3407854963');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CA98333DA', 'Giulia', 'Castro', 'F', 'GLICST95L10F839K', TO_DATE('10-09-1995', 'DD-MM-YYYY'), 'Via aniello Falcone', '96A', 80070, '3336544152');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CE23534DW', 'Chiara', 'Verdi', 'F', 'CHRVRD99P05F839B', TO_DATE('05-12-1999', 'DD-MM-YYYY'), 'Via alessandro manzoni', '74', 80045, '3898554212');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CX98547DF', 'Mario', 'Paese', 'M', 'MRAPSA98H20F839U', TO_DATE('20-11-1998', 'DD-MM-YYYY'), 'Corso arnaldo lucci', '98', 80024, '3207856478');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CF93120UP', 'Pietro', 'Danti', 'M', 'PTRDNT80J30F839F', TO_DATE('30-09-1980', 'DD-MM-YYYY'), 'Via battistello caracciolo', '1', 80050, '3921485678');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CS98756LK', 'Maria', 'Improta', 'F', 'MRAIPT84D17F839Q', TO_DATE('17-12-1984', 'DD-MM-YYYY'), 'Vico nocelle', '11', 80059, '3421265983');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CQ98765IU', 'Giuseppe', 'Tramonto', 'M', 'GSPTMN94D08F839Z', TO_DATE('08-05-1994', 'DD-MM-YYYY'), 'Piazza carlo III', '8', 80020, '3887596341');
INSERT INTO Persona(numero_documento_persona, nome_persona, cognome_persona, genere_persona, codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona, telefono_persona) VALUES('CW49856DV', 'Carla', 'Giallo', 'F', 'CRLGLL62U04F839F', TO_DATE('04-01-1962', 'DD-MM-YYYY'), 'Via foria', '30', 80139, '3886574896');
```

```
-- Popolamento Tipologia turno –
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Mattutino', 6.00, 12.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Pomeridiano', 12.00, 18.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Serale', 18.00, 23.30);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Part-time Matt', 8.00, 12.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Part-time Pome', 15.00, 19.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Full-time Matt', 7.00, 15.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Full-time Pome', 12.00, 20.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Full-time Sera', 16.00, 24.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Matt + Pome', 9.00, 17.00);
INSERT INTO Tipologia_turno(nome_tipologia_turno, ora_inizio_turno, ora_fine_turno) VALUES('Pome esteso', 13.00, 19.00);

-- Popolamento Esercizio –
INSERT INTO Esercizio VALUES ('Tapis roulant', 'Cardio');
INSERT INTO Esercizio VALUES ('Spinte chest press', 'Pettorali');
INSERT INTO Esercizio VALUES ('Spinte shoulder press', 'Spalle');
INSERT INTO Esercizio VALUES ('Pectoral Machine', 'Pettorali');
INSERT INTO Esercizio VALUES ('Tirate cavi basso', 'Tricipiti');
INSERT INTO Esercizio VALUES ('French press su panca', 'Tricipiti');
INSERT INTO Esercizio VALUES ('Abdominal Machine', 'Addome');
INSERT INTO Esercizio VALUES ('Alzate su panca reclinata', 'Addome');
INSERT INTO Esercizio VALUES ('Leg extension', 'Quadricipiti');
INSERT INTO Esercizio VALUES ('Leg curl seduto', 'Femorali');
INSERT INTO Esercizio VALUES ('Slanci gamba con multi hip', 'Glutei');
INSERT INTO Esercizio VALUES ('Arnold press', 'Spalle');
INSERT INTO Esercizio VALUES ('Alzate laterali', 'Spalle');

-- Popolamento Prodotto –
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1553257892411', 'Acqua naturale Vera', 0.9,
555);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1594231692412', 'Acqua frizzante Ferrarelle',
0.9, 550);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1598257892413', 'Monster', 2.59, 150);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('3141243492414', 'Tè verde', 1.5, 120);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1598657522415', 'Energaid', 2.0, 60);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('8565387952416', 'Gatoraid', 2.0, 70);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1594257892417', 'Barretta energetica', 3.0,
25);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('5252547525418', 'Barretta proteica', 3.5, 30);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1598648972419', 'Acqua naturale panna', 1.5,
10);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('1598645826420', 'TUC', 0.6, 250);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('5936578924214', 'Crackers', 0.5, 300);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('3132157892422', 'Croccantelle prosciutto
crudo', 0.7, 550);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('7578657892423', 'Croccantelle bacon', 0.7,
550);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('7744765692424', 'Croccantelle pomodoro',
0.7, 550);
INSERT INTO Prodotto(codice_a_barre, nome_prodotto, prezzo_vendita_unitario, giacenza) VALUES('7896553546425', 'Budino proteico', 3.29, 47);

-- Popolamento Corso –
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Sala', 'Allenamento');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Pump', 'Definizione');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Total body', 'Massa');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Crossfit', 'Rafforzamento');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Yoga', 'Armonizzazione corpo');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('GAG', 'Glutei e gambe');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Zumba', 'Divertimento');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('TRX', 'Tonificazione corpo');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Kick boxing', 'Difesa personale');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Step', 'Forma fisica');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Nuoto', 'Definizione');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Karate', 'Difesa personale');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Pilates', 'Zona inferiore');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Aerobica', 'Metabolismo');
INSERT INTO Corso(nome_corso, obiettivo) VALUES('Spinning', 'Preparazione fisica');
```

-- Popolamento Tipologia Abbonamento –

```
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Sala', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Pump', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Total body', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Crossfit', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Yoga', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo GAG', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Zumba', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo TRX', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Kick boxing', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Step', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Nuoto', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Karate', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Pilates', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Aerobica', 1, 39.99);
INSERT INTO Tipologia_abbonamento(nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento) VALUES('Solo Spinning', 1, 39.99);
```

-- Popolamento Dipendente –

```
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78534DV', 'IT125478965874589628478548 5', 'Istruttore', 900, 'Scienze Motorie', 'Via salvator rosa', '150', 80136);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CB98765VF', 'IT125478965154826598478548 5', 'Segretaria', 1000, 'Via Mar Rosso', '5', 24123);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78144DV', 'IT125478961480640407604478 0', 'Istruttore', 900, 'Scienze Motorie', 'Via salvator rosa', '150', 80136);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CU36584LB', 'IT120569870364589615904784 2', 'Istruttore', 1200, 'Scienze Motorie', 'Via Giuseppe Verdi', '1', 80121);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78734DV', 'IT125478965871457480126048 5', 'Responsabile', 1500, 'Economia', 'Via Napoli', '3', 24126);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA95534DV', 'IT114507565835120480518 5485', 'Responsabile', 1300, 'Ingegneria gestionale', 'Via salvator rosa', '150', 80136);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CD35795MN', 'IT250145405874501234052678 0', 'Responsabile', 1200, 'Scienze Motorie', 'Via Mar Rosso', '5', 24123);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CF36945GV', 'IT110405565874501654061651 2', 'Istruttore', 900, 'Scienze Motorie', 'Piazza Garibaldi', '33', 80136);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CG36854JK', 'IT121054868674512305564201 0', 'Istruttore', 1000, 'Scienze Motorie', 'Viale Manzoni', '53', 80121);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CJ32478PL', 'IT122206165871334027486045 8', 'Responsabile', 1300, 'Scienze Motorie', 'Viale Manzoni', '53', 80121);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CP14567DF', 'IT101453265874581678704680 4', 'Responsabile', 1200, 'Scienze Motorie', 'Via Corso Europa', '17', 80016);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CM14587KF', 'IT104565048848945634486343 4', 'Responsabile', 1250, 'Scienze Motorie', 'Via Giuseppe Verdi', '1', 80121);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA70034DV', 'IT162504859845404940548048 1', 'Segretaria', 900, 'Scienze Motorie', 'Via salvator rosa', '150', 80136);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CO11025RE', 'IT125478965406894085499640 6', 'Istruttore', 900, 'Scienze Motorie', 'Via Giuseppe Verdi', '1', 80121);
INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile, titolo_di_studio, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CB98564FR', 'IT125478965874580589164808 8', 'Istruttore', 900, 'Scienze Motorie', 'Via salvator rosa', '150', 80136);
```

```
-- Popolamento Utente –
INSERT INTO Utente(fk_persona) VALUES('CA78534DV');
INSERT INTO Utente(fk_persona) VALUES('CA78534DK');
INSERT INTO Utente(fk_persona) VALUES('CA78548DL');
INSERT INTO Utente(fk_persona) VALUES('CA79834DZ');
INSERT INTO Utente(fk_persona) VALUES('CA98333DA');
INSERT INTO Utente(fk_persona) VALUES('CE23534DW');
INSERT INTO Utente(fk_persona) VALUES('CX98547DF');
INSERT INTO Utente(fk_persona) VALUES('CF93120UP');
INSERT INTO Utente(fk_persona) VALUES('CS98756LK');
INSERT INTO Utente(fk_persona) VALUES('CQ98765IU');
INSERT INTO Utente(fk_persona) VALUES('CW49856DV');
INSERT INTO Utente(fk_persona) VALUES('CP36578JD');
INSERT INTO Utente(fk_persona) VALUES('CR98736CD');
INSERT INTO Utente(fk_persona) VALUES('CO01536KS');
INSERT INTO Utente(fk_persona) VALUES('CR02459NC');

-- Popolamento Utenze –
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('1SA5FE78DS', 'Via salvator rosa', '150', 80136, 150.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ENEL-LUCE');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('OLFDSPK145', 'Via salvator rosa', '150', 80136, 250.99, TO_DATE('30-06-23', 'DD-MM-YY'),  'ENEL-ELETTRICITA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('TKUKI21YTV', 'Via salvator rosa', '150', 80136, 135.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ACQUA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('ZBYHJT20YJ', 'Via Mar Rosso', '5', 24123, 160.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ENEL-LUCE');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('JYUT02NHKJ', 'Via Mar Rosso', '5', 24123, 85.99, TO_DATE('30-06-23', 'DD-MM-YY'),  'ENEL-ELETTRICITA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('HTR12SD5JY', 'Via Mar Rosso', '5', 24123, 167.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ACQUA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('AFE02RG31R', 'Via Giuseppe Verdi', '1', 80121, 260.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ENEL-LUCE');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('1EFW12GRH0', 'Via Giuseppe Verdi', '1', 80121, 365.99, TO_DATE('30-06-23', 'DD-MM-YY'),  'ENEL-ELETTRICITA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('HT0R2HD3BT', 'Via Giuseppe Verdi', '1', 80121, 150.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ACQUA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('THR21HGF1T', 'Piazza Garibaldi', '33', 80136, 190.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ENEL-LUCE');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('1NH0122YT0', 'Piazza Garibaldi', '33', 80136, 185.99, TO_DATE('30-06-23', 'DD-MM-YY'),  'ENEL-ELETTRICITA');
INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza, tipologia_utenza)
VALUES('78DSGRE1RE', 'Piazza Garibaldi', '33', 80136, 141.99, TO_DATE('30-06-23', 'DD-MM-YY'), 'ACQUA');

-- Popolamento Presenza dipendente –
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('15-06-23 6:05', 'DD-MM-YY HH24:MI'), to_timestamp('15-06-23 12:03', 'DD-MM-YYYY HH24:MI'), 'CA78534DV', 'Via
salvator rosa', '150', 80136, 'Mattutino');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('16-06-23 12:01', 'DD-MM-YY HH24:MI'), to_timestamp('16-06-23 18:03', 'DD-MM-YYYY HH24:MI'), 'CB98765VF', 'Via Mar
Rosso', '5', 24123, 'Pomeridiano');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('17-06-23 18:25', 'DD-MM-YY HH24:MI'), to_timestamp('17-06-23 23:01', 'DD-MM-YYYY HH24:MI'), 'CA78144DV', 'Via
salvator rosa', '150', 80136, 'Serale');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('19-06-23 8:00', 'DD-MM-YY HH24:MI'), to_timestamp('15-06-23 12:05', 'DD-MM-YYYY HH24:MI'), 'CU36584LB', 'Via
Giuseppe Verdi', '1', 80121, 'Part-time Matt');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('15-06-23 15:30', 'DD-MM-YY HH24:MI'), to_timestamp('15-06-23 19:05', 'DD-MM-YYYY HH24:MI'), 'CA78534DV', 'Via
Napoli', '3', 24126, 'Part-time Pome');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('17-06-23 7:00', 'DD-MM-YY HH24:MI'), to_timestamp('17-06-23 15:05', 'DD-MM-YYYY HH24:MI'), 'CB98765VF', 'Via Mar
```

```
Rosso', '5', 24123, 'Full-time Matt');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('16-06-23 12:40', 'DD-MM-YY HH24:MI'), to_timestamp('16-06-23 20:10', 'DD-MM-YYYY HH24:MI'), 'CA78144DV', 'Via
salvator rosa', '150', 80136, 'Full-time Pome');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('20-06-23 16:05', 'DD-MM-YY HH24:MI'), to_timestamp('21-06-23 00:05', 'DD-MM-YYYY HH24:MI'), 'CF36945GV', 'Piazza
Garibaldi', '33', 80136, 'Full-time Sera');
INSERT INTO Presenza_dipendente(data_ora_entrata, data_ora_uscita, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
fk_tipologia_turno_nome)
  VALUES (to_timestamp('17-06-23 6:05', 'DD-MM-YY HH24:MI'), to_timestamp('17-06-23 12:03', 'DD-MM-YYYY HH24:MI'), 'CG36854JK', 'Viale
Manzoni', '53', 80121, 'Matt + Pome');

-- Popolamento Scheda di allenamento –
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('26-06-23', 'DD-MM-YY'),
'CA78534DV', 'CF36945GV', TO_DATE('26-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('28-06-23', 'DD-MM-YY'),
'CA78534DK', 'CB98564FR', TO_DATE('28-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('01-07-23', 'DD-MM-YY'),
'CA78548DL', 'CG36854JK', TO_DATE('28-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('25-06-23', 'DD-MM-YY'),
'CA79834DZ', 'CB98564FR', TO_DATE('28-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('29-06-23', 'DD-MM-YY'),
'CA98333DA', 'CB98564FR', TO_DATE('01-08-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('28-06-23', 'DD-MM-YY'),
'CE23534DW', 'CF36945GV', TO_DATE('30-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('15-07-23', 'DD-MM-YY'),
'CX98547DF', 'CG36854JK', TO_DATE('31-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('03-07-23', 'DD-MM-YY'),
'CF93120UP', 'CF36945GV', TO_DATE('23-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('28-06-23', 'DD-MM-YY'),
'CS98756LK', 'CF36945GV', TO_DATE('15-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('05-07-23', 'DD-MM-YY'),
'CQ98765IU', 'CF36945GV', TO_DATE('05-08-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('06-07-23', 'DD-MM-YY'),
'CW49856DV', 'CF36945GV', TO_DATE('26-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('28-06-23', 'DD-MM-YY'),
'CP36578JD', 'CF36945GV', TO_DATE('28-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('30-06-23', 'DD-MM-YY'),
'CR98736CD', 'CG36854JK', TO_DATE('15-07-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('29-06-23', 'DD-MM-YY'),
'CO01536KS', 'CF36945GV', TO_DATE('29-08-23', 'DD-MM-YY'));
INSERT INTO Scheda_di_allenamento(data_inizio_scheda, fk_utente, fk_dipendente, data_fine_scheda) VALUES (TO_DATE('30-06-23', 'DD-MM-YY'),
'CR02459NC', 'CG36854JK', TO_DATE('28-08-23', 'DD-MM-YY'));

-- Popolamento Scontrino –
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Mar Rosso', '5', 24123, 0.90);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Viale Manzoni', '53', 80121, 0.90);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Viale delle industrie', '22', 80142, 6);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Viale Manzoni', '53', 80121, 3.99);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Massimo', '76', 61126, 1.20);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Massimo', '76', 61126, 3.29);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Rosa', '4', 24122, 2.50);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Piazza Verdi', '50', 29132, 5.50);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Viale delle industrie', '22', 80142, 1.50);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Napoli', '3', 24126, 1.50);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Napoli', '3', 24126, 1.00);
INSERT INTO Scontrino(fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale) VALUES ('Via Massimo', '76', 61126, 2.0);
```

```
-- Popolamento Piscina –
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(1, 'Via Mar Rosso', '5', 24123, 25, 50.5, 2, 15, 8);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, capienza) VALUES(2, 'Via Mar Rosso', '5', 24123, 15);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(3, 'Via Mar Rosso', '5', 24123, 20, 45, 2.5, 15, 7);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(1, 'Via Giuseppe Verdi', '1', 80121, 25, 50.5, 2, 15, 8);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, capienza) VALUES(2, 'Via Giuseppe Verdi', '1', 80121, 15);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(3, 'Via Giuseppe Verdi', '1', 80121, 20, 45, 2.5, 15, 7);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(1, 'Piazza Garibaldi', '33', 80136, 25, 50.5, 2, 15, 8);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, capienza) VALUES(2, 'Piazza Garibaldi', '33', 80136, 15);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(3, 'Piazza Garibaldi', '33', 80136, 10, 25, 1.5, 15, 3);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(1, 'Viale Manzoni', '53', 80121, 15, 30.5, 2, 15, 4);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, capienza) VALUES(2, 'Viale Manzoni', '53', 80121, 15);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, larghezza, lunghezza ,profondita ,capienza, numero_corsie)
VALUES(1, 'Via Corso Europa', '17', 80016, 15, 30.5, 2, 15, 4);
INSERT INTO Piscina(numero_piscina, fk_sede_via, fk_sede_civico, fk_sede_cap, capienza) VALUES(2, 'Via Corso Europa', '17', 80016, 15);

-- Popolamento Sala –
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('A1', 'Via Mar Rosso', '5', 24123,
30, 'Attrezzi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('A2', 'Via Mar Rosso', '5', 24123,
25, 'Fitness');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('A3', 'Via Mar Rosso', '5', 24123,
25, 'Corsi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('B1', 'Via Giuseppe Verdi', '1',
80121, 30, 'Attrezzi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('B2', 'Via Giuseppe Verdi', '1',
80121, 25, 'Fitness');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('B3', 'Via Giuseppe Verdi', '1',
80121, 25, 'Corsi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('C1', 'Piazza Garibaldi', '33',
80136, 30, 'Attrezzi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('C2', 'Piazza Garibaldi', '33',
80136, 25, 'Fitness');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('C3', 'Piazza Garibaldi', '33',
80136, 25, 'Corsi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('D1', 'Viale Manzoni', '53',
80121, 30, 'Attrezzi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('D2', 'Viale Manzoni', '53',
80121, 25, 'Fitness');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('D3', 'Viale Manzoni', '53',
80121, 25, 'Corsi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('F1', 'Via Corso Europa', '17',
80016, 30, 'Attrezzi');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('F2', 'Via Corso Europa', '17',
80016, 25, 'Fitness');
INSERT INTO Sala(codice_sala, fk_sede_via, fk_sede_civico, fk_sede_cap, metri_quadri_sala, tipologia_sala) VALUES('F3', 'Via Corso Europa', '17',
80016, 25, 'Corsi');
```

-- Popolamento Sottoscrizione –

```
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78534DV', 'Solo Sala', 'Via Mar Rosso', '5', 24123);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78534DV', 'Anaerobico', 'Via Mar Rosso', '5', 24123);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78534DV', 'Solo Nuoto', 'Via Mar Rosso', '5', 24123);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78534DK', 'Solo Sala', 'Via Giuseppe Verdi', '1', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78548DL', 'Sala + Anaerobico', 'Via Giuseppe Verdi', '1', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA78548DL', 'Solo Nuoto', 'Via Giuseppe Verdi', '1', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA79834DZ', 'Solo Sala', 'Piazza Garibaldi', '33', 80136);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CA98333DA', 'Solo Sala', 'Viale Manzoni', '53', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CE23534DW', 'Sala + Difesa', 'Viale Manzoni', '53', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CE23534DW', 'Anaerobico', 'Viale Manzoni', '53', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CE23534DW', 'Solo GAG', 'Viale Manzoni', '53', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CX98547DF', 'Solo Nuoto', 'Viale Manzoni', '53', 80121);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CF93120UP', 'Solo Sala', 'Via Corso Europa', '17', 80016);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CF93120UP', 'Solo Kick boxing', 'Via Corso Europa', '17', 80016);
INSERT INTO Sottoscrizione(fk_utente, fk_abbonamento_nome, fk_sede_via, fk_sede_civico, fk_sede_cap) VALUES('CF93120UP', 'Solo Spinning', 'Via Corso Europa', '17', 80016);
```

-- Popolamento Assegnazione settimanale –

```
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('15-06-23', 'DD-MM-YY'), 'CA78534DV', 'Mattutino');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('16-06-23', 'DD-MM-YY'), 'CB98765VF', 'Pomeridiano');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('17-06-23', 'DD-MM-YY'), 'CA78144DV', 'Serale');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('19-06-23', 'DD-MM-YY'), 'CU36584LB', 'Part-time Matt');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('15-06-23', 'DD-MM-YY'), 'CA78534DV', 'Part-time Pome');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('17-06-23', 'DD-MM-YY'), 'CB98765VF', 'Full-time Matt');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('16-06-23', 'DD-MM-YY'), 'CA78144DV', 'Full-time Pome');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('20-06-23', 'DD-MM-YY'), 'CF36945GV', 'Full-time Sera');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('17-06-23', 'DD-MM-YY'), 'CG36854JK', 'Matt + Pome');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('15-06-23', 'DD-MM-YY'), 'CF36945GV', 'Pome esteso');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('16-06-23', 'DD-MM-YY'), 'CG36854JK', 'Pome esteso');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('20-06-23', 'DD-MM-YY'), 'CG36854JK', 'Full-time Pome');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('22-06-23', 'DD-MM-YY'), 'CA70034DV', 'Mattutino');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('21-06-23', 'DD-MM-YY'), 'CO11025RE', 'Serale');
INSERT INTO Assegnazione_settimanale(data_assegnazione, fk_dipendente, fk_tipologia_turno_nome) VALUES(TO_DATE('22-06-23', 'DD-MM-YY'), 'CB98564FR', 'Matt + Pome');
```

-- Popolamento Lezione corso –

```
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('D1', 'Viale Manzoni', '53', 80121, 'Karate', 'CG36854JK', 'FRI', 14.00, 15.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('B1', 'Via Giuseppe Verdi', '1', 80121, 'Yoga', 'CU36584LB', 'MON', 9.00, 10.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('C2', 'Piazza Garibaldi', '33', 80136, 'Zumba', 'CF36945GV', 'TUE', 18.00, 19.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('B3', 'Via Giuseppe Verdi', '1', 80121, 'Aerobica', 'CO11025RE', 'WED', 19.00, 20.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('B3', 'Via Giuseppe Verdi', '1', 80121, 'Spinning', 'CO11025RE', 'WED', 20.00, 21.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('C3', 'Piazza Garibaldi', '33', 80136, 'Step', 'CF36945GV', 'THU', 14.00, 15.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('B3', 'Via Giuseppe Verdi', '1', 80121, 'Aerobica', 'CO11025RE', 'WED', 20.00, 21.00);
INSERT INTO Lezione_corso(fk_sala_codice, fk_sala_via, fk_sala_civico, fk_sala_cap, fk_corso, fk_dipendente, giorno_settimana_lezione, ora_inizio, ora_fine) VALUES('D1', 'Viale Manzoni', '53', 80121, 'Pump', 'CG36854JK', 'FRI', 16.00, 17.00);
```

-- Popolamento Contiene –

```
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Tapis roulant', 0, 0, 0);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Spinte chest press', 50, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(2, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Spinte chest press', 50, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(3, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Spinte chest press', 50, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(4, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Spinte chest press',  50, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Tirate cavi basso',  25, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(2, TO_DATE('26-06-23', 'DD-MM-YY'), 'CA78534DV', 'Tirate cavi basso',  25, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('28-06-23', 'DD-MM-YY'), 'CA78534DK', 'Tapis roulant', 0, 0, 0);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('28-06-23', 'DD-MM-YY'), 'CA78534DK', 'Pectoral Machine', 35, 12, 30);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('28-06-23', 'DD-MM-YY'), 'CA78534DK', 'Tirate cavi basso', 20, 12, 30);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('28-06-23', 'DD-MM-YY'), 'CA78534DK', 'Abdominal Machine', 0, 15, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('01-07-23', 'DD-MM-YY'), 'CA78548DL', 'Tapis roulant', 0, 0, 0);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('01-07-23', 'DD-MM-YY'), 'CA78548DL', 'Pulley', 60, 8, 80);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('01-07-23', 'DD-MM-YY'), 'CA78548DL', 'Lat machine avanti', 75, 8, 80);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('01-07-23', 'DD-MM-YY'), 'CA78548DL', 'Curl hammer con cavi', 14, 8, 80);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('25-06-23', 'DD-MM-YY'), 'CA79834DZ', 'Tapis roulant', 0, 0, 0);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('25-06-23', 'DD-MM-YY'), 'CA79834DZ', 'Leg press', 100, 10, 60);
INSERT INTO Contiene(numero_serie, fk_scheda_data, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni, secondi_di_recupero)
  VALUES(1, TO_DATE('25-06-23', 'DD-MM-YY'), 'CA79834DZ', 'Stacchi a gambe tese con bilanciere', 20, 10, 60);
```

-- Popolamento Vendita –

```
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('1', '1553257892411', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('2', '1594231692412', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('3', '1594257892417', 2);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('4', '1598257892413', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('4', '3132157892422', 2);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('5', '1598645826420', 2);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('6', '7896553546425', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('7', '8565387952416', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('7', '5936578924214', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('9', '1598648972419', 1);
```

```
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('10', '3141243492414', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('11', '5936578924214', 2);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('12', '8565387952416', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('21', '5252547525418', 1);
INSERT INTO Vendita(fk_scontrino, fk_prodotto, quantita_venduta) VALUES('21', '1598657522415', 1);
```

-- Popolamento Prenotazione –

```
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('26-06-23 10:00', 'DD-MM-YY HH24:MI'), 'CA78534DK', 1, 'Via Mar Rosso', '5', 24123, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('27-06-23 10:00', 'DD-MM-YY HH24:MI'), 'CA78534DK', 1, 'Via Mar Rosso', '5', 24123, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('28-06-23 10:00', 'DD-MM-YY HH24:MI'), 'CA78534DV', 3, 'Via Mar Rosso', '5', 24123, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('29-06-23 10:00', 'DD-MM-YY HH24:MI'), 'CA78534DK', 1, 'Via Giuseppe Verdi', '1', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('01-07-23 10:00', 'DD-MM-YY HH24:MI'), 'CA98333DA', 1, 'Via Giuseppe Verdi', '1', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('03-07-23 10:00', 'DD-MM-YY HH24:MI'), 'CA78534DV', 3, 'Via Mar Rosso', '5', 24123, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('04-07-23 10:00', 'DD-MM-YY HH24:MI'), 'CA98333DA', 2, 'Via Giuseppe Verdi', '1', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('06-07-23 10:00', 'DD-MM-YY HH24:MI'), 'CA78534DV', 2, 'Via Giuseppe Verdi', '1', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('26-06-23 10:00', 'DD-MM-YY HH24:MI'), 'CA98333DA', 2, 'Via Giuseppe Verdi', '1', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('26-06-23 12:00', 'DD-MM-YY HH24:MI'), 'CA78534DV', 1, 'Via Giuseppe Verdi', '1', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('26-06-23 14:00', 'DD-MM-YY HH24:MI'), 'CS98756LK', 1, 'Viale Manzoni', '53', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('28-06-23 11:00', 'DD-MM-YY HH24:MI'), 'CR98736CD', 1, 'Viale Manzoni', '53', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('26-06-23 10:30', 'DD-MM-YY HH24:MI'), 'CX98547DF', 1, 'Viale Manzoni', '53', 80121, 20.00);
INSERT INTO Prenotazione(data_ora_prenotazione, fk_utente, fk_piscina_numero, fk_piscina_via, fk_piscina_civico, fk_piscina_cap,
prezzo_prenotazione) VALUES(TO_TIMESTAMP('28-06-23 10:00', 'DD-MM-YY HH24:MI'), 'CR02459NC', 2, 'Via Corso Europa', '17', 80016, 20.00);
```

-- Popolamento Compreso –

```
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Sala', 'Solo Sala', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Pump', 'Solo Pump', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Total body', 'Solo Total body', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Crossfit', 'Solo Crossfit', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Yoga', 'Solo Yoga', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('GAG', 'Solo GAG', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Zumba', 'Solo Zumba', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('TRX', 'Solo TRX', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Kick boxing', 'Solo Kick boxing', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Nuoto', 'Solo Nuoto', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Karate', 'Solo Karate', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Pilates', 'Solo Pilates', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Step', 'Solo Step', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Aerobica', 'Solo Aerobica', 1);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Sala', 'Sala + Anaerobico', 3);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Pump', 'Sala + Anaerobico', 3);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('TRX', 'Sala + Anaerobico', 3);
INSERT INTO Compreso(fk_corso, fk_abbonamento_nome, fk_abbonamento_durata) VALUES('Total body', 'Sala + Anaerobico', 3);
```

7.3 – Triggers

I trigger DML sono utili principalmente per il controllo di vincoli dinamici in fase di immissione o aggiornamento dei dati. È possibile usarli anche per implementare regole di business o per calcolare, generare o sovrascrivere il valore di alcuni campi.

Controllo capienza piscina

Questo trigger ha lo scopo, prima dell'inserimento all'interno della tabella Prenotazione, di controllare se nella fascia oraria della prenotazione ci sia effettivamente disponibilità per prenotare la piscina, nel caso di uno di questi errori viene lanciato un messaggio di errore personalizzato.

```
-- CHECK CAPIENZA PISCINA --
CREATE OR REPLACE TRIGGER check_capienza_piscina
BEFORE INSERT ON Prenotazione
FOR EACH ROW
DECLARE
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui ci fossero troppo prenotazioni per una piscina, per la quale si è superata la capienza massima --
  capienza_exc EXCEPTION;

  -- VARIABILI --
  numero_persone NUMBER := 0;
  capienza_piscina NUMBER := 0;

BEGIN
  -- Contiamo quanti utenti hanno prenotato la piscina nell'arco di -1 ora e +59 minuti a partire dalla nuova prenotazione che si vuole effettuare --
  SELECT COUNT(fk_utente) INTO numero_persone
    FROM Prenotazione WHERE data_ora_prenotazione
      BETWEEN (:NEW.data_ora_prenotazione - INTERVAL '1' HOUR) AND (:NEW.data_ora_prenotazione + INTERVAL '59' MINUTE)
      AND fk_piscina_numero = :NEW.fk_piscina_numero AND fk_piscina_via = :NEW.fk_piscina_via
      AND fk_piscina_cap = :NEW.fk_piscina_cap AND fk_piscina_civico = :NEW.fk_piscina_civico;

  -- Reperiamo l'informazione riguardo la capienza della piscina --
  SELECT capienza INTO capienza_piscina FROM Piscina
    WHERE numero_piscina = :NEW.fk_piscina_numero AND fk_sede_via = :NEW.fk_piscina_via
    AND fk_sede_cap = :NEW.fk_piscina_cap AND fk_sede_civico = :NEW.fk_piscina_civico;

  -- Controlliamo che le prenotazioni non superino la capienza massima della piscina --
  IF (numero_persone = capienza_piscina) THEN
    RAISE capienza_exc;
  END IF;

EXCEPTION
  WHEN capienza_exc THEN
    RAISE_APPLICATION_ERROR('-20112', 'È STATA RAGGIUNTA GIÀ LA CAPIENZA MASSIMA PER QUELLA PISCINA NELLA DATA E ORA: ' || :NEW.data_ora_prenotazione);

END;
```

Controllo date schede di allenamento

Questo trigger ha lo scopo, prima dell'inserimento di una scheda di allenamento, di controllare se la data di inizio sia antecedente al giorno e maggiore della data fine e che quest'ultima non sia minore di 7 giorni dalla data di inizio, nel caso di uno di questi errori viene lanciato un messaggio di errore personalizzato.

```
--CHECK DATA SCHEDA--
CREATE OR REPLACE TRIGGER check_date_scheda
BEFORE INSERT OR UPDATE OF data_fine_scheda ON scheda_di_allenamento
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui la data dell'inizio della scheda di allenamento risultasse minore della data
odierna --
   data_inizio_scheda_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui la data della fine della scheda di allenamento risultasse minore della scadenza
minima --
   -- Ovvero, minore della (data inizio + 7), in questo si è deciso che una scheda deve durare almeno 7 giorni --
   data_fine_scheda_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui la data dell'inizio della scheda di allenamento risultasse maggiore della data
della fine della
   --scheda
  data_inizio_maggiore_fine_exc EXCEPTION;

   -- VARIABILI --
  data_scadenza_minima scheda_di_allenamento.data_fine_scheda%TYPE := :new.data_inizio_scheda + 7;

BEGIN
   -- Controlliamo se la data di inizio è maggiore della data di fine –
   IF ( :new.data_inizio_scheda > :new.data_fine_scheda ) THEN
      RAISE data_inizio_maggiore_fine_exc;
   END IF;

   -- Controlliamo se la data di inizio è minore della data odierna --
   IF ( :new.data_inizio_scheda < trunc(sysdate) ) THEN
      RAISE data_inizio_scheda_exc;
   END IF;

   -- Controlliamo se la data di fine è minore della scadenza minima (data inizio + 7) --
   IF ( :new.data_fine_scheda < data_scadenza_minima ) THEN
       RAISE data_fine_scheda_exc;
   END IF;

EXCEPTION
   WHEN data_inizio_scheda_exc THEN
      raise_application_error('-20100', '(SCHEDA_DI_ALLENAMENTO) ERRORE "data_inizio_scheda": '
                              || TO_DATE(:new.data_inizio_scheda, 'DD/MM/YYYY')
                              || ' < '
                              || sysdate);

   WHEN data_fine_scheda_exc THEN
      raise_application_error('-20101', '(SCHEDA_DI_ALLENAMENTO) ERRORE "data_fine_scheda": '
                              || TO_DATE(:new.data_fine_scheda, 'DD/MM/YYYY')
                              || ' < '
                              || TO_DATE(data_scadenza_minima, 'DD/MM/YYYY'));

   WHEN data_inizio_maggiore_fine_exc THEN
      raise_application_error('-20102', '(SCHEDA_DI_ALLENAMENTO) ERRORE "data_inizio_scheda" MAGGIORE "data_fine_scheda":
'
                              || TO_DATE(:new.data_inizio_scheda, 'DD/MM/YYYY')
                              || ' > '
```

```
                    || TO_DATE(:new.data_fine_scheda, 'DD/MM/YYYY'));
END;
```

Controllo data prenotazione

Questo trigger ha lo scopo, prima dell'inserimento di una prenotazione, di controllare se la data di quest'ultima è antecedente al giorno ed ora attuale, nel caso viene lanciato un messaggio di errore personalizzato.

```
--CONTROLLO DATA PRENOTAZIONE--
CREATE OR REPLACE TRIGGER chk_data_prenotazione
BEFORE INSERT ON Prenotazione
FOR EACH ROW
DECLARE
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui la data della prenotazione risultasse minore della data odierna –
  date_exc EXCEPTION;

BEGIN
  -- Controlliamo se la data della prenotazione è minore della data odierna --
  IF (:NEW.data_ora_prenotazione < CURRENT_TIMESTAMP) THEN
    RAISE date_exc;
  END IF;

EXCEPTION
  WHEN date_exc THEN
    RAISE_APPLICATION_ERROR('-20110', 'DATA ED ORA INIZIO MINORI DI: ' || CURRENT_TIMESTAMP);
END;

--CHECK DATA UTENZA--
CREATE OR REPLACE TRIGGER chk_data_utenza
BEFORE INSERT ON Utenza
FOR EACH ROW
DECLARE
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui la data scadenza dell'utenza risultasse minore della data odiera --
  data_scadenza_exc EXCEPTION;

BEGIN
  -- Controlliamo se la data scadenza da inserire è minore o uguale alla data odierna --
  IF(:NEW.data_scadenza <= SYSDATE) THEN
    RAISE data_scadenza_exc;
  END IF;

EXCEPTION
  WHEN data_scadenza_exc THEN
   RAISE_APPLICATION_ERROR('-20108', 'LA DATA SCADENZA ' || :NEW.data_scadenza || ' < ' || SYSDATE);
END;
```

Controllo dipendente lezione corso

Questo trigger ha lo scopo, prima dell'inserimento di una lezione corso, di controllare se il dipendente lavora nella sede dove si deve tenere la lezione, poi se il dipendente è un istruttore e non è stato licenziato e se è stato assegnato in quelle ore per poter effettivamente tenere la lezione del corso, nel caso di uno di questi errori viene lanciato un messaggio di errore personalizzato.

```
-- CHECK INSERIMENTO LEZIONE --
CREATE OR REPLACE TRIGGER chk_inserimento_lezione
BEFORE INSERT ON Lezione_corso
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO" --
   sede_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui a compilare la scheda sia un dipendente che non è un istruttore --
   mansione_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui si vuole assegnare ad una lezione un dipendente che non è di turno --
   dip_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui non si può effettuare una lezione nella sala selezionata --
   sala_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui il corso non prevede lezioni --
   corso_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui la sede inserita non coincide con la sede del dipendente --
   sede_dip_exc EXCEPTION;

    -- VARIABILI --
  stato_sedeX Sede.stato_sede%type;
   tipo_sala Sala.tipologia_sala%type;
   corso_nome Corso.nome_corso%type;
   dipendente_info Dipendente%rowtype;
   counter_dip NUMBER := 0;

 BEGIN
  -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
   SELECT stato_sede INTO stato_sedeX FROM Sede
      WHERE via_sede = :NEW.fk_sala_via AND civico_sede = :NEW.fk_sala_civico AND cap_sede = :NEW.fk_sala_cap;

    IF (stato_sedeX = 'NON ATTIVO') THEN
      RAISE sede_exc;
   END IF;

    -- Reperiamo la mansione del dipendente che vuole compilare la scheda e controlliamo se non è licenziato --
  SELECT * INTO dipendente_info FROM dipendente WHERE fk_persona = :NEW.fk_dipendente AND data_di_licenziamento IS
NULL;

   -- Controlliamo se la mansione del dipendente è diversa da istruttore oppure null e controlliamo se la sede in cui --
   -- viene ospitata la lezione è diversa da quella in cui lavora il dipendente --
   IF(UPPER(dipendente_info.mansione) != 'ISTRUTTORE' OR dipendente_info.mansione IS NULL) THEN
      RAISE mansione_exc;
   ELSIF(dipendente_info.fk_sede_via != :NEW.fk_sala_via AND dipendente_info.fk_sede_civico != :NEW.fk_sala_civico
            AND dipendente_info.fk_sede_cap != :NEW.fk_sala_cap) THEN
       RAISE sede_dip_exc;
   END IF;

   -- Contiamo quante assegnazioni turno della settimana prossima ha il dipendente durante la fascia oraria della lezione che si
vuole inserire --
   SELECT count(fk_dipendente) INTO counter_dip FROM Assegnazione_settimanale JOIN Tipologia_turno
      ON fk_tipologia_turno_nome = nome_tipologia_turno
      WHERE fk_dipendente = :NEW.fk_dipendente AND TO_CHAR( data_assegnazione, 'DY' ) = :NEW.giorno_settimana_lezione
      AND data_assegnazione BETWEEN TRUNC(SYSDATE + 7,'D') + 1 AND TRUNC(SYSDATE + 7,'D') + 6
```

```
    AND (:NEW.ora_inizio BETWEEN ora_inizio_turno AND ora_fine_turno
    AND :NEW.ora_fine BETWEEN ora_inizio_turno AND ora_fine_turno);

  -- Se il dipendente non è stato assegnato durante le ore della lezione, allora significa che durante quelle ore –
  -- il dipendente non è di turno --
  IF(counter_dip = 0) THEN
      RAISE dip_exc;
  END IF;

  -- Se la sala corrisponde alla sala attrezzi, non sarà possibile svolgere una lezione al suo interno --
  SELECT tipologia_sala INTO tipo_sala FROM Sala
    WHERE fk_sede_via = :NEW.fk_sala_via AND fk_sede_civico = :NEW.fk_sala_civico AND fk_sede_cap = :NEW.fk_sala_cap
            AND codice_sala =  :NEW.fk_sala_codice;

  IF(UPPER(tipo_sala) = 'ATTREZZI') THEN
     RAISE sala_exc;
  END IF;

 -- Il corso "SALA" non è un corso che prevede lezioni, in quanto autorizza solo l'utilizzo della sala attrezzi --
 SELECT nome_corso INTO corso_nome FROM Corso
    WHERE nome_corso=:NEW.fk_corso;

 IF(UPPER(corso_nome) = 'SALA') THEN
    RAISE corso_exc;
 END IF;

EXCEPTION
  WHEN sede_exc THEN
    RAISE_APPLICATION_ERROR('-20121', 'LA SEDE NON È ATTIVA');

  WHEN mansione_exc THEN
    RAISE_APPLICATION_ERROR('-20122', 'LA MANSIONE DEL DIPENDENTE È DIVERSA DA ISTRUTTORE: ' ||
dipendente_info.mansione);

  WHEN dip_exc THEN
    RAISE_APPLICATION_ERROR('-20123', 'IL DIPENDENTE ' || :NEW.fk_dipendente || ' NON È ASSEGNATO DURATE QUELLE ORE');

  WHEN sala_exc THEN
    RAISE_APPLICATION_ERROR('-20124', 'NON È POSSIBILE SVOLGERE LEZIONI NELLA SALA ' || tipo_sala);

 WHEN corso_exc THEN
    RAISE_APPLICATION_ERROR('-20125', 'IL CORSO ' || corso_nome || ' NON È UN CORSO CHE PREVEDE LEZIONI');

  WHEN sede_dip_exc THEN
    RAISE_APPLICATION_ERROR('-20125', 'IL DIPENDENTE NON LAVORA IN QUESTA SEDE');
END;
```

Controllo numero lezioni giornaliere

Questo trigger ha lo scopo, prima dell'inserimento di una lezione, di controllare se si è già raggiunto il limite massimo (5) di lezioni giornaliere per quella sede, nel caso viene lanciato un messaggio di errore personalizzato.

```
--CONTROLLO NUMERO LEZIONI GIORNALIERE--
CREATE OR REPLACE TRIGGER chk_numero_lezione_giornaliere
BEFORE INSERT ON Lezione_corso
FOR EACH ROW
DECLARE
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui venissero inserite più di 5 lezioni giornaliere per sede --
  lezioni_giornaliere_exc EXCEPTION;

  -- VARIABILI --
  numero_lezioni NUMBER := 0;

BEGIN
  -- Contiamo quante lezioni sono disponibili nella sala in uno specifico giorno --
  SELECT COUNT(id_lezione) INTO numero_lezioni
    FROM Lezione_corso
    WHERE giorno_settimana_lezione = :NEW.giorno_settimana_lezione AND fk_sala_via = :NEW.fk_sala_via
    AND fk_sala_cap = :NEW.fk_sala_cap AND fk_sala_civico = :NEW.fk_sala_civico;

  -- Se ci fossero più di 5 lezioni, allora lanciamo l'eccezione --
  IF (numero_lezioni >= 5) THEN
    RAISE lezioni_giornaliere_exc;
  END IF;

EXCEPTION
  WHEN lezioni_giornaliere_exc THEN
    RAISE_APPLICATION_ERROR('-20111', 'CI SONO GIÀ 5 LEZIONI PER QESTO GIORNO');
END;
```

Controllo uscita dipendente

Questo trigger ha lo scopo, prima della modifica della presenza del dipendente (quindi nel caso in cui il dipendente voglia uscire), di controllare se il dipendente sta effettuando l'uscita all'orario della fine del suo turno, in caso negativo viene lanciato un messaggio di errore personalizzato.

```
--CHK PRESENZA--
CREATE OR REPLACE TRIGGER chk_presenza
BEFORE UPDATE OF data_ora_uscita ON Presenza_dipendente
FOR EACH ROW
DECLARE
  -- EXCEPTION --
  -- Questa eccezione viene lanciata nel caso in cui il dipendente tenti di registrare un'uscita prima della fine del proprio turno --
  uscita_exc EXCEPTION;

  -- VARIABILI --
  ora_curr_timeX NUMBER := 0;
  ora_fine_turnoX NUMBER := 0;

BEGIN
  -- Reperiamo l'informazione relativa alla ora di fine turno del dipendente --
  SELECT ora_fine_turno INTO ora_fine_turnoX FROM Tipologia_turno WHERE nome_tipologia_turno =
:NEW.fk_tipologia_turno_nome;

  -- Salviamo in una variabile l'ora attuale --
 ora_curr_timeX := to_number(to_char(CURRENT_TIMESTAMP, 'HH24.MI'));
```

```
  -- Controlliamo se l'ora attuale risulta minore dell'orario di uscita del dipendente --
  IF(ora_curr_timeX < ora_fine_turnoX) THEN
    RAISE uscita_exc;
  END IF;

EXCEPTION
  WHEN uscita_exc THEN
    RAISE_APPLICATION_ERROR('-20117', 'USCITA NON CONSENTITA');
END;
```

Controllo sottoscrizione

Questo trigger ha lo scopo, prima dell'inserimento di una sottoscrizione, di controllare se l'utente ha già effettuato una sottoscrizione a una tipologia abbonamento che comprende uno dei corsi nuovi a cui si vuole sottoscrivere l'utente o se uno dei corsi ha raggiunto il limite massimo (75) di persone per quel corso in quella sede, in uno dei due casi viene lanciato un messaggio di errore personalizzato.

```
--CHK_SOTTOSCRIZIONE--
CREATE OR REPLACE TRIGGER chk_sottoscrizione
BEFORE INSERT ON Sottoscrizione
FOR EACH ROW
DECLARE
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui l'utente tenti di sottoscriversi ad una tipologia abbonamento --
  -- che comprende un corso a cui è già sottoscritto --
  sottoscrizione_exc EXCEPTION;
  -- Questa eccezione viene lanciata nel caso in cui si raggiungesse il limite massimo di iscritti ad un corso --
  max_utenti_exc EXCEPTION;

  -- CURSORI --
  -- Selezioniamo tutti i corsi compresi in una tipologia_abbonamento --
  CURSOR C
  IS
  SELECT fk_corso AS nome_c FROM Compreso WHERE fk_abbonamento_nome = :NEW.fk_abbonamento_nome;

  -- VARIABILI --
  utenti_corsoX NUMBER := 0;
  count_corsi NUMBER := 0;
  durata_abbonamento_sot Tipologia_abbonamento.durata_abbonamento%TYPE;

BEGIN
  -- Per ogni corso selezionato, contiamo quanti utenti, di una specifica sede, sono iscritti e hanno pagato --
  -- Se il count raggiungesse il numero 75, allora abbiamo raggiunto il limite di sottoscrizioni per quel corso --
  FOR rec IN C
  LOOP
    SELECT count(fk_utente) INTO utenti_corsoX FROM sottoscrizione JOIN compreso
      ON sottoscrizione.fk_abbonamento_nome = compreso.fk_abbonamento_nome
      WHERE fk_corso = rec.nome_c AND fk_sede_via = :NEW.fk_sede_via
      AND fk_sede_cap = :NEW.fk_sede_cap AND fk_sede_civico = :NEW.fk_sede_civico
      AND ha_pagato = 'S';

    IF (utenti_corsoX = 75) THEN
      RAISE max_utenti_exc;
    END IF;
  END LOOP;

  -- Contiamo quante volte appare il corso nei corsi compresi nella sottoscrizione dell'utente --
  -- Se fosse maggiore di 0, allora vuol dire che l'utente si vuole iscrivere ad una tipologia --
  -- abbonamento che comprende un corso a cui è già iscritto --
  SELECT count(fk_corso) INTO count_corsi FROM Compreso WHERE fk_corso
```

```
   IN (SELECT fk_corso FROM Sottoscrizione JOIN Compreso
            ON Sottoscrizione.fk_abbonamento_nome = Compreso.fk_abbonamento_nome
            WHERE fk_utente = :NEW.fk_utente GROUP BY fk_corso)
   AND fk_abbonamento_nome = :NEW.fk_abbonamento_nome;

   IF(count_corsi > 0) THEN
      RAISE sottoscrizione_exc;
   END IF;

   -- Reperiamo la durata dell'abbonamento a cui l'utente vuole sottoscriversi --
   SELECT durata_abbonamento INTO durata_abbonamento_sot FROM Tipologia_abbonamento
      WHERE nome_tipologia_abbonamento = :NEW.fk_abbonamento_nome;

   -- Inseriamo i dati relativi alla durata della sottoscrizione e la data fine sottoscrizione, ottenuta tramite la durata --
   -- più la data di oggi --
   :NEW.fk_abbonamento_durata := durata_abbonamento_sot;
   :NEW.data_fine_sottoscrizione := ADD_MONTHS(:NEW.data_inizio_sottoscrizione, durata_abbonamento_sot);

EXCEPTION
   WHEN max_utenti_exc THEN
      RAISE_APPLICATION_ERROR('-20113', 'NUMERO MASSIMO DI UTENTI ISCRITTI ');

   WHEN sottoscrizione_exc THEN
            RAISE_APPLICATION_ERROR('-20116', 'LA NUOVA SOTTOSCRIZIONE COMPRENDE UN CORSO GIÀ SOTTOSCRITTO
DALL''UTENTE ' || :NEW.fk_utente);
END;

--PAGAMENTO DELLA SOTTOSCRIZIONE--
CREATE OR REPLACE TRIGGER chk_sottoscrizione_pagata
BEFORE UPDATE OF ha_pagato ON Sottoscrizione
FOR EACH ROW
BEGIN
   -- In caso di rinnovo dell'iscrizione, cambiamo i valori della data nuova data inizio sottoscrizione --
   -- alla data odierna e la nuova data fine sottoscrizione
   IF(:NEW.ha_pagato = 'S' AND :OLD.ha_pagato = 'N') THEN
      :NEW.data_inizio_sottoscrizione := SYSDATE;
      :NEW.data_fine_sottoscrizione := add_months(SYSDATE, :NEW.fk_abbonamento_durata);
   END IF;
END;
```

Controllo sovrapposizione turni dipendente

Questo trigger ha lo scopo, prima dell'inserimento di un'assegnazione settimanale, di controllare se questa dovesse sovrapporsi con una già assegnata allo stesso dipendente, nel caso viene lanciato un messaggio di errore personalizzato.

```
--CONTROLLO ASSEGNAZIONE TURNI SOVRAPPOSTI--
CREATE OR REPLACE TRIGGER chk_turni
BEFORE INSERT ON Assegnazione_settimanale
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui si vuole assegnare il dipendente ad un turno --
   -- in cui gli orari sono compresi in un turno a cui è stato già assegnato --
   turno_exc EXCEPTION;

   -- VARIABILI --
   ora_inizio_turnoX NUMBER;
   ora_fine_turnoX NUMBER;
   count_dipendente NUMBER := 0;
```

```
BEGIN
   -- Ricaviamo le informazioni relativi all'ora inizio turno e ora fine turno --
   SELECT ora_inizio_turno, ora_fine_turno INTO ora_inizio_turnoX, ora_fine_turnoX FROM Tipologia_turno
      WHERE nome_tipologia_turno = :NEW.fk_tipologia_turno_nome;

   -- Contiamo se il dipendente è stato già assegnato per quella data e per quella fascia oraria --
   SELECT count(fk_dipendente) INTO count_dipendente FROM Assegnazione_settimanale JOIN Tipologia_turno
      ON fk_tipologia_turno_nome = nome_tipologia_turno
      WHERE data_assegnazione = :NEW.data_assegnazione AND fk_dipendente = :NEW.fk_dipendente
      AND (ora_inizio_turnoX BETWEEN ora_inizio_turno AND ora_fine_turno
            OR ora_fine_turnoX BETWEEN ora_inizio_turno AND ora_fine_turno);

   -- Se il count è maggiore di 0 significa che l'utente ha già un'assegnazione disponibile per quella fascia oraria --
   IF(count_dipendente > 0) THEN
      RAISE turno_exc;
   END IF;

EXCEPTION
   WHEN turno_exc THEN
            RAISE_APPLICATION_ERROR('-20118', 'IL DIPENDENTE È GIÀ STATO ASSEGNATO PER QUESTE ORE');
END;
```

Controllo prenotazione utente

Questo trigger ha lo scopo, prima dell'inserimento di una prenotazione, di controllare se l'utente ha già effettuato una prenotazione per quella piscina nell'arco temporale di 1h (tempo della prenotazione) sia prima che dopo la nuova data di prenotazione, nel caso viene lanciato un messaggio di errore personalizzato.

```
--CHK_UTENTE_PRENOTATO--
CREATE OR REPLACE TRIGGER chk_utente_prenotato
BEFORE INSERT ON PRENOTAZIONE
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui l'utente tenti di prenotarsi negli orari compresi in una sua prenotazione già
effettuata --
   -- ora prima di una sua prenotazione, in quanto non vogliamo che l'utente possa sovrapporre gli orari di prenotazione --
   prenotazione_exc EXCEPTION;

   -- VARIABILI --
   count_prenotazione NUMBER := 0;

BEGIN
   -- Contiamo quante prenotazioni ha già effettuato l'utente in quella piscina che partono dall'orario selezionato -1 ora e +1 ora --
   SELECT COUNT(fk_utente) INTO count_prenotazione FROM Prenotazione
      WHERE fk_utente = :NEW.fk_utente
   AND data_ora_prenotazione BETWEEN (:NEW.data_ora_prenotazione - INTERVAL '1' HOUR)
                                    AND (:NEW.data_ora_prenotazione + INTERVAL '1' HOUR)
   AND fk_piscina_numero = :NEW.fk_piscina_numero AND fk_piscina_via = :NEW.fk_piscina_via
   AND fk_piscina_cap = :NEW.fk_piscina_cap AND fk_piscina_civico = :NEW.fk_piscina_civico;

   -- Se è presente una prenotazione, significa che l'utente potrebbe sovrapporre la prenotazione ad una già effettuata --
   IF (count_prenotazione > 0) THEN
      RAISE prenotazione_exc;
   END IF;

EXCEPTION
   WHEN prenotazione_exc THEN
      RAISE_APPLICATION_ERROR('-20114', 'L'' UTENTE HA GIÀ EFFETTUATO LA PRENOTAZIONE NELLA STESSA PISCINA NELL''ARCO
DI 1H');
END;
```

Controllo tipologia dipendete

Questo trigger ha lo scopo, prima dell'inserimento di un dipendente, di controllare in primis se l'età del dipendente è maggiore di 18, poi, di controllare se la sede alla quale lo si vuole assegnare ha già raggiunto il limite massimo di dipendenti in totale o specifici per la mansione del nuovo dipendente e nel caso viene stampato un messaggio di errore personalizzato.

```
--CONTROLLO SUL DIPENDENTE--
CREATE OR REPLACE TRIGGER genera_dipendente
BEFORE INSERT ON Dipendente
FOR EACH ROW
DECLARE
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui il dipendente sia minorenne, impendendo il suo inserimento --
  eta_exc EXCEPTION;
  -- Questa eccezione viene lanciata nel caso in cui si fosse raggiunto il massimo delle mansioni consentite per sede --
  tipo_dip_exc EXCEPTION;
  -- Questa eccezione viene lanciata nel caso in cui si fosse raggiunto il limite di dipendenti per sede --
  tot_dip_exc EXCEPTION;

  -- VARIABILI --
  eta_dip NUMBER;
  count_dip NUMBER :=0;

BEGIN
  -- Ricaviamo l'età del dipendente. Se essa fosse minore di 18, lanciamo l'eccezione --
  SELECT eta_persona INTO eta_dip FROM Persona WHERE numero_documento_persona = :NEW.fk_persona;

  IF (eta_dip < 18) THEN
    RAISE eta_exc;
  END IF;

  -- Contiamo quante mansioni dello stesso tipo sono già presenti nella sede in cui vogliamo inserire il nuovo dipendente --
  -- Se fossero già presenti 3 istruttori, allora impendiamo l'inserimento --
  -- Se fossero già presenti 2 segretarie, allora impendiamo l'inserimento --
  -- Se fossero già presente 1 responsabile, allora impendiamo l'inserimento --
  SELECT COUNT(mansione) INTO count_dip FROM Dipendente
    WHERE mansione = :NEW.mansione AND fk_sede_via = :NEW.fk_sede_via
      AND fk_sede_civico = :NEW.fk_sede_civico AND fk_sede_cap = :NEW.fk_sede_cap;

  IF(UPPER(:NEW.mansione) = 'ISTRUTTORE' AND count_dip = 3) THEN
    RAISE tipo_dip_exc;
  ELSIF (UPPER(:NEW.mansione) = 'SEGRETARIA' AND count_dip = 2) THEN
    RAISE tipo_dip_exc;
  ELSIF (UPPER(:NEW.mansione) = 'RESPONSABILE' AND count_dip = 1) THEN
    RAISE tipo_dip_exc;
  END IF;

  -- Contiamo quanti dipendenti sono stati assunti nella sede dove si vuole inserire il nuovo dipendente --
  -- Se ci fossero 6 dipendenti, allora la sede è piena e impendiamo l'inserimento --
  SELECT COUNT(fk_persona) INTO count_dip FROM Dipendente
    WHERE fk_sede_via = :NEW.fk_sede_via AND fk_sede_civico = :NEW.fk_sede_civico
      AND fk_sede_cap = :NEW.fk_sede_cap;

  IF(count_dip = 6) THEN
    DELETE FROM Persona WHERE numero_documento_persona = :NEW.fk_persona;
    RAISE tot_dip_exc;
  END IF;
```

```
EXCEPTION
   WHEN eta_exc THEN
      RAISE_APPLICATION_ERROR('-20104', 'ETA DEL DIPENDENTE MINORE DI 18 ');

   WHEN tipo_dip_exc THEN
      RAISE_APPLICATION_ERROR('-20105', 'CI SONO GIÀ ' || count_dip ||' '|| :NEW.mansione|| ' NELLA SEDE SELEZIONATA');

   WHEN tot_dip_exc THEN
      RAISE_APPLICATION_ERROR('-20106', 'CI SONO GIÀ 6 NELLA SEDE SELEZIONATA');
END;
```

Età persona

Questo trigger ha lo scopo di calcolare l'età e assegnarla alla nuova persona che si vuol inserire. Prima dell'assegnazione avviene anche il controllo che l'età sia compresa tra l'età minima e quella massima accettate.

```
--CALCOLO E CONTROLLO ETA--
CREATE OR REPLACE TRIGGER genera_eta_persona
BEFORE INSERT ON Persona
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui l'utente non soddisfa i requisiti di età minima e massima per iscriversi --
   eta_exc EXCEPTION;

   -- VARIABILI --
   eta_massima NUMBER := 100;
   eta_minima NUMBER := 6;
   eta NUMBER;

BEGIN
   -- Otteniamo, dalla data di nascita della persona, la sua età --
   eta := TO_NUMBER ( months_between(trunc(sysdate), :NEW.data_di_nascita) / 12 );

   -- Controlliamo se non è compresa tra l'età minima e l'età massima per impedire l'inserimento della persona --
   IF (eta NOT BETWEEN eta_minima AND eta_massima) THEN
      RAISE eta_exc;
   END IF;

   -- Inseriamo l'età della persona --
   :NEW.eta_persona := eta;

EXCEPTION
   WHEN eta_exc THEN
      RAISE_APPLICATION_ERROR('-20103', 'ETA NON CONSENTITA PER L''ISCRIZIONE ALLA PALESTRA');
END;
```

Auto-incremento id lezione corso

Questo trigger ha lo scopo di incrementare il valore della sequenza id_lezione_seq per poterlo poi assegnare, come id lezione, nel momento della registrazione, di una nuova lezione corso.

```
--ID LEZIONE CORSO--
CREATE OR REPLACE TRIGGER id_lezione_trigger
BEFORE INSERT ON lezione_corso
FOR EACH ROW
BEGIN
   -- Inseriamo come nuovo id lezione il valore successivo della sequenza --
   :NEW.id_lezione := id_lezione_seq.NEXTVAL;
END;
```

Auto-incremento id scontrino

Questo trigger ha lo scopo di incrementare il valore della sequenza id_scontrino_seq per poterlo poi assegnare, come id scontrino, nel momento della registrazione, di un nuovo scontrino.

```
--ID SCONTRINO SCONTRINO--
CREATE OR REPLACE TRIGGER id_scontrino_trigger
BEFORE INSERT ON scontrino
FOR EACH ROW
BEGIN
   -- Inseriamo come nuovo id scontrino il valore successivo della sequenza --
   :NEW.id_scontrino := id_scontrino_seq.NEXTVAL;
END;
```

Auto-incremento id sottoscrizione

Questo trigger ha lo scopo di incrementare il valore della sequenza id_sottoscrizione_seq per poterlo poi assegnare, come id sottoscrizione, nel momento della registrazione, di una nuova sottoscrizione.

```
--ID SOTTOSCRIZIONE UTENTE--
CREATE OR REPLACE TRIGGER id_sottoscrizione_trigger
BEFORE INSERT ON sottoscrizione
FOR EACH ROW
BEGIN
   -- Inseriamo come nuovo id sottoscrizione il valore successivo della sequenza --
   :NEW.id_sottoscrizione := id_sottoscrizione_seq.NEXTVAL;
END;
```

Auto-incremento numero tessera utente

Questo trigger ha lo scopo di incrementare il valore della sequenza numero_tessera_seq per poterlo poi assegnare, come numero tessera, nel momento della registrazione, di un nuovo utente.

```
--NUMERO TESSERA UTENTE--
CREATE OR REPLACE TRIGGER numero_tessera_trigger
BEFORE INSERT ON Utente
FOR EACH ROW
BEGIN
   -- Inseriamo come nuovo numero tessera il valore successivo della sequenza --
   :NEW.numero_tessera := numero_tessera_seq.NEXTVAL;
END;
```

Max ore settimanali

Questo trigger ha lo scopo, prima dell'inserimento di un'assegnazione settimanale, di controllare se il dipendente ha già raggiunto o sta superando, con la nuova assegnazione, il numero massimo di ore lavorative settimanali previste (48h) e nel caso stampare un messaggio di errore personalizzato.

```
--MAX_ORE_SETTIMANALI--
CREATE OR REPLACE TRIGGER max_ore_settimanali
BEFORE INSERT ON Assegnazione_settimanale
FOR EACH ROW
DECLARE
      -- ECCEZIONI --
      -- Questa eccezione viene lanciata nel caso in cui il dipendente avesse superato le 48 ore settimanali --
      ore_sett_exc EXCEPTION;

       -- VARIABILI --
       ore_turno_ass NUMBER := 0;
       ore_nuova_ass NUMBER := 0;

BEGIN
        -- Sommiamo le ore di lavoro settimanali del dipendente
        SELECT SUM(ora_fine_turno - ora_inizio_turno) INTO ore_turno_ass
        FROM Tipologia_turno JOIN Assegnazione_settimanale
        ON nome_tipologia_turno = fk_tipologia_turno_nome
        WHERE fk_dipendente = :NEW.fk_dipendente;



        -- Selezioniamo le ore di lavoro del dipendente nel nuovo turno che si vorrebbe aggiungere --
        SELECT (ora_fine_turno - ora_inizio_turno) INTO ore_nuova_ass FROM Tipologia_turno
            WHERE nome_tipologia_turno = :NEW.fk_tipologia_turno_nome;

        -- Se la somma tra le ore di lavoro assegnate e le nuove ore ottenute dal nuovo assegnamento --
        -- superano le 48 ore settimanali, lanciamo l'errore --
        IF( (ore_turno_ass + ore_nuova_ass) > 48) THEN
            RAISE ore_sett_exc;
        END IF;

EXCEPTION
WHEN ore_sett_exc THEN
        RAISE_APPLICATION_ERROR('-20119','ERRORE MAX ORE SETTIMANALI RAGGIUNTE PER QUESTO DIPENDENTE');
END;
```

Controllo giacenza prodotti

Questo trigger ha lo scopo di controllare, prima dell'inserimento della vendita, se la quantità che si vuole vendere è maggiore della giacenza del prodotto, nel caso viene mostrato un messaggio di errore personalizzato.

```
--MAX PRODOTTI VENDUTI--
CREATE OR REPLACE TRIGGER max_prodotti_venduti
BEFORE INSERT ON VENDITA
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui vengano venduti più prodotti di quanti ne siano disponibili in giacenza --
   giacenza_exc EXCEPTION;

   -- VARIABILI --
   giacenzaX Prodotto.giacenza%type;
```

```
BEGIN
   -- Reperiamo la giacenza dei prodotti --
   SELECT giacenza INTO giacenzaX FROM Prodotto WHERE codice_a_barre = :NEW.fk_prodotto;

   -- Controlliamo se la giacenza è minore della quantità che si vuole vendere --
   IF (giacenzaX < :NEW.quantita_venduta) THEN
      RAISE giacenza_exc;
    END IF;

EXCEPTION
   WHEN giacenza_exc THEN
         RAISE_APPLICATION_ERROR('-20109', 'LA GIACENZA DEL PRODOTTO ' || :NEW.fk_prodotto || ' È MINORE DELLA QUANTITA
CHE SI VUOL VENDERE, GIACENZA: ' || giacenzaX);
END;
```

Sede non attiva

Questo trigger ha lo scopo, prima della modifica di una sede da Attivo a NON Attivo, di controllare se ci sono ancora utenze pendenti da dover pagare, in caso affermativo viene stampato un messaggio di errore personalizzato.

```
--SEDE NON ATTIVA--
CREATE OR REPLACE TRIGGER sede_non_attiva
BEFORE UPDATE OF stato_sede ON Sede
FOR EACH ROW
DECLARE
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui ci fossero ancora delle utenze da pagare nella sede che si vuole dichiare nello
stato di "NON ATTIVO"
   utenze_exc EXCEPTION;

   -- VARIABILI --
   chk_utenze NUMBER := 0;

BEGIN
   -- Contiamo quante utenze ci sono ancora da pagare nella sede che si vuole cambiare lo stato in "NON ATTIVO" --
   -- Se fosse maggiore di 0, allora vanno pagate delle utenze e non si può aggiornare la sede --
   SELECT COUNT(numero_fattura) INTO chk_utenze FROM Utenza  WHERE fk_sede_via = :NEW.via_sede
         AND fk_sede_civico = :NEW.civico_sede AND fk_sede_cap = :NEW.cap_sede AND pagamento_utenza = 'N';

   IF (chk_utenze > 0) THEN
      RAISE utenze_exc;
   END IF;

   IF(UPPER(:NEW.stato_sede) = 'NON ATTIVO') THEN
      UPDATE Dipendente SET data_di_licenziamento = sysdate WHERE fk_sede_via = :NEW.via_sede
         AND fk_sede_civico = :NEW.civico_sede AND fk_sede_cap = :NEW.cap_sede;
   END IF;

EXCEPTION
      WHEN utenze_exc THEN
         RAISE_APPLICATION_ERROR('-20107', 'ESISTONO ANCORA DELLE UTENZE DA PAGARE');
END;
```

7.4 – Procedure e funzioni

Le procedure sono la parte più strettamente legata alla logica di business e all'automazione della Base di Dati. Se nel DDL e nel DML il problema principale è decidere la rappresentazione ottima dei dati e garantire rispetto dei vincoli di integrità su di essi, in questa fase il problema è come sfruttare al meglio tale rappresentazione per automatizzarne la gestione e trarne il massimo profitto.

Assegnazione dipendente

Questa procedura, ha come argomenti le informazioni relative ad un'assegnazione settimanale: il numero documento del dipendente, la data della settimana successiva e il nome della tipologia turno da assegnare al dipendente. Viene controllato se per caso si vuole inserire un dipendente licenziato, poi la validità della data per vedere se fa parte della prossima settimana ed infine viene inserita l'assegnazione settimanale.

```
CREATE OR REPLACE PROCEDURE assegna_dipendente(numero_dipendente VARCHAR2,  data_assegnazione VARCHAR2,
                                                nome_tipologia_turno VARCHAR2)
AS
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui la data inserita in input non appartiene ad una data --
   -- della settimana successiva a partire dalla settimana corrente
   data_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui il dipendente risulta licenziato --
   licenziato_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui il dipendente sia un responsabile --
   responsabile_exc EXCEPTION;

   -- VARIABILI --
  data_assegnazioneX DATE := to_date(data_assegnazione, 'DD-MM-YYYY');
  data_lic DATE;
  mansione_dip Dipendente.mansione%type;

BEGIN
   -- Controlliamo che il dipendente non sia licenziato e non sia un responsabile --
   SELECT data_di_licenziamento, mansione INTO data_lic, mansione_dip FROM dipendente
        WHERE fk_persona = numero_dipendente;

   IF (data_lic IS NULL) THEN
      RAISE licenziato_exc;
   ELSIF(UPPER(mansione_dip) = 'RESPONSABILE') THEN
            RAISE responsabile_exc;
   END IF;

   -- Controlliamo la validità della data inserita in input. Se essa non appartiene alla settimana successiva --
   -- vuol dire che si sta cercando di assegnare il dipendente per un'altra settimana e non per quella successiva --
   -- alla data di oggi --
   IF(data_assegnazioneX NOT BETWEEN TRUNC(SYSDATE + 7,'D') + 1 AND TRUNC(SYSDATE + 7,'D') + 6) THEN
      RAISE data_exc;
   END IF;

   -- Inseriamo i dati relativi all'assegnazione settimanale del dipendente --
   INSERT INTO Assegnazione_settimanale VALUES (data_assegnazioneX, numero_dipendente, nome_tipologia_turno);

EXCEPTION
   WHEN data_exc THEN
      RAISE_APPLICATION_ERROR('-20011', 'DATA ASSEGNAZIONE NON COMPRESA NELL''ARCO TEMPORALE DELLA PROSSIMA
SETTIMANA');

   WHEN licenziato_exc THEN
      RAISE_APPLICATION_ERROR('-20012', 'DIPENDENTE LICENZIATO');
```

```
  WHEN responsabile_exc THEN
    RAISE_APPLICATION_ERROR('-20013', 'IL DIPENDENTE È UN RESPONSABILE');
END;
```

Assunzione

Questa procedura prende come argomenti le informazioni della persona e la sede in cui verrà chiamata a svolgere le proprie mansioni. Questa procedura ha un duplice utilizzo: assumere o aggiornare. Se il dipendente già esiste, tramite questa procedura si aggiornano le informazioni della tabella dipendente. Se non esiste verrà assunto.

```
CREATE OR REPLACE PROCEDURE assunzione(numero_persona CHAR,  via_sedeX  VARCHAR2,  civico_sedeX VARCHAR2,
                                       cap_sedeX  NUMBER,  codice_fiscale CHAR,  data_di_nascita VARCHAR2,
                                       codice_ibanX CHAR,  mansioneX  VARCHAR2,  nome VARCHAR2 := NULL,
                                      cognome VARCHAR2 := NULL,  genere  CHAR := NULL,  via_persona  VARCHAR2 := NULL,
                                       civico_persona    VARCHAR2 := NULL,  cap_persona NUMBER := NULL,
                                  telefono  CHAR := NULL,  stipendioX  NUMBER := NULL,  titolo_di_studioX VARCHAR2 :=NULL)
  AS
    -- ECCEZIONI --
    -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO" --
    sede_exc EXCEPTION;

    -- VARIABILI --
    chk_dipendente NUMBER;
    data_di_nascitaX DATE := TO_DATE ( data_di_nascita, 'DD-MM-YYYY' );
    stato_sedeX Sede.stato_sede%type;

  BEGIN
    -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
    SELECT stato_sede INTO stato_sedeX FROM Sede
      WHERE via_sede = via_sedeX AND civico_sede = civico_sedeX AND cap_sede = cap_sedeX;

    IF (stato_sedeX = 'NON ATTIVO') THEN
      RAISE sede_exc;
    END IF;

    -- Controlliamo se già esiste un dipendente all'interno della tabella "Dipendente" con il numero del documento passato in input
    SELECT COUNT(fk_persona) INTO chk_dipendente FROM Dipendente WHERE fk_persona = numero_persona;

    -- Se esiste, allora si è fatta richiesta di aggiornare i dati relativi al dipendente, in particolare: --
    -- 1. Un possibile cambio di sede --
    -- 2. Un cambio di stipendio --
    -- 3. Una riassunzione --
    -- Infatti, se lo stipendio fosse a NULL, allora si sta cercando di cambiare i dati relativi a 1. e 3. --
    -- Caso in cui lo stipendo fosse diverso da NULL, allora si sta cercando di effettuare un aumento e, probabilmente, la 1. e la 3. --
    IF (chk_dipendente = 1) THEN
      IF(stipendioX IS NULL) THEN
        UPDATE Dipendente SET fk_sede_via = via_sedeX, fk_sede_civico = civico_sedeX, fk_sede_cap = cap_sedeX,
                              data_di_licenziamento = NULL,  stipendio_mensile = DEFAULT
        WHERE fk_persona = numero_persona;
      ELSE
        UPDATE Dipendente SET fk_sede_via = via_sedeX, fk_sede_civico = civico_sedeX, fk_sede_cap = cap_sedeX,
                              data_di_licenziamento = NULL,  stipendio_mensile = stipendioX
        WHERE fk_persona = numero_persona;
      END IF;
    ELSE
      -- Se il dipendente non esiste, allora si tratta di un'assunzione. Per cui inseriamo i dati relativi alla persona --
      -- e successivamente al dipendente --
      INSERT INTO persona (numero_documento_persona, nome_persona, cognome_persona, genere_persona,
                           codice_fiscale_persona, data_di_nascita, via_persona, civico_persona, cap_persona,
telefono_persona)
        VALUES (numero_persona, nome, cognome, genere, codice_fiscale, data_di_nascitaX, via_persona, civico_persona,
```

```
                cap_persona, telefono);


        IF(stipendioX IS NULL) THEN
          INSERT INTO Dipendente(fk_persona, codice_iban, mansione,  titolo_di_studio, fk_sede_via, fk_sede_civico,
fk_sede_cap)
            VALUES(numero_persona, codice_ibanX, mansioneX, titolo_di_studioX, via_sedeX, civico_sedeX, cap_sedeX);
        ELSE
          INSERT INTO Dipendente(fk_persona, codice_iban, mansione, stipendio_mensile,  titolo_di_studio, fk_sede_via,
                                 fk_sede_civico, fk_sede_cap)
          VALUES(numero_persona, codice_ibanX, mansioneX, stipendioX, titolo_di_studioX, via_sedeX, civico_sedeX, cap_sedeX);
      END IF;
 END IF;

EXCEPTION
   WHEN sede_exc THEN
     RAISE_APPLICATION_ERROR('-20007', 'LA SEDE NON È ATTIVA');
END;
```

Compila scheda

Questa procedura permette ad un istruttore di compilare una scheda di allenamento per un utente della palestra. La scheda può contenere un massimo di 9 esercizi, dove il primo è obbligatorio per la creazione. Il dipendente deve essere per forza un istruttore e l'utente deve essere iscritto al corso di "sala"

```
--COMPILA SCHEDA ALLENAMENTO--

CREATE OR REPLACE PROCEDURE compila_scheda( data_inizio_scheda VARCHAR2, data_fine_scheda VARCHAR2,
                                            fk_scheda_utente CHAR, fk_dipendente CHAR, numero_serie NUMBER,
                                            fk_esercizio VARCHAR2, peso NUMBER, numero_ripetizioni NUMBER,
                                            secondi_di_recupero NUMBER, numero_serie2 NUMBER := NULL,
                                            fk_esercizio2 VARCHAR2 := NULL,  peso2 NUMBER := NULL,
                                            numero_ripetizioni2 NUMBER := NULL,  secondi_di_recupero2 NUMBER := NULL,
                                           numero_serie3 NUMBER := NULL, fk_esercizio3 VARCHAR2 := NULL,
                                           peso3 NUMBER := NULL, numero_ripetizioni3 NUMBER := NULL,
                                           secondi_di_recupero3 NUMBER := NULL, numero_serie4 NUMBER := NULL,
                                           fk_esercizio4 VARCHAR2 := NULL,  peso4 NUMBER := NULL,
                                          numero_ripetizioni4 NUMBER := NULL, secondi_di_recupero4 NUMBER := NULL,
                                          numero_serie5 NUMBER := NULL, fk_esercizio5 VARCHAR2 := NULL,
                                          peso5 NUMBER := NULL, numero_ripetizioni5 NUMBER := NULL,
                                           secondi_di_recupero5 NUMBER := NULL, numero_serie6 NUMBER := NULL,
                                           fk_esercizio6 VARCHAR2 := NULL,  peso6 NUMBER := NULL,
                                           numero_ripetizioni6 NUMBER := NULL, secondi_di_recupero6 NUMBER := NULL,
                                           numero_serie7 NUMBER := NULL,  fk_esercizio7 VARCHAR2 := NULL,
                                           peso7 NUMBER := NULL, numero_ripetizioni7 NUMBER := NULL,
                                           secondi_di_recupero7 NUMBER := NULL, numero_serie8 NUMBER := NULL,
                                           fk_esercizio8 VARCHAR2 := NULL, peso8 NUMBER := NULL,
                                           numero_ripetizioni8 NUMBER := NULL,  secondi_di_recupero8 NUMBER := NULL,
                                           numero_serie9 NUMBER := NULL, fk_esercizio9 VARCHAR2 := NULL,
                                           peso9 NUMBER := NULL, numero_ripetizioni9 NUMBER := NULL,
                                          secondi_di_recupero9 NUMBER := NULL)
  AS
     -- ECCEZIONI --
     -- Questa eccezione viene lanciata nel caso in cui a compilare la scheda sia un dipendente che non è un istruttore --
     mansione_exc EXCEPTION;

    -- VARIABILI --
    mansione_dip Dipendente.mansione%type;
    abbonamento_sala_utente Compreso%rowtype;
    data_inizio_schedaX DATE := TO_DATE ( data_inizio_scheda, 'DD-MM-YY' );
    data_fine_schedaX DATE := TO_DATE ( data_fine_scheda, 'DD-MM-YY' );
```

```
  BEGIN
    -- Effettuiamo un controllo sulla sottoscrizione dell'utente che comprenda il corso "sala" --
    -- La query più interna restituisce il nome degli abbonamenti dell'utente che sono stati pagati e che abbiano il certificato
medico consegnato --
    -- La query esterna filtra dal risultato della query interna solo i corsi con il nome di "sala" --
    -- Se non esistesse, verrebbe lanciato un errore, impedendo il proseguimento della procedura --
    SELECT * INTO abbonamento_sala_utente FROM Compreso WHERE Compreso.fk_abbonamento_nome
    IN (SELECT fk_abbonamento_nome FROM Sottoscrizione WHERE fk_utente = fk_scheda_utente
              AND UPPER(ha_pagato) = 'S' AND UPPER(certificato_medico) = 'S')
    AND LOWER(fk_corso) = 'sala';

   - - Reperiamo la mansione del dipendente che vuole compilare la scheda e controlliamo se non è licenziato --
    SELECT mansione INTO mansione_dip FROM dipendente
      WHERE fk_persona = fk_dipendente AND data_di_licenziamento IS NULL;

    -- Controlliamo se la mansione del dipendente è diversa da istruttore oppure null, per lanciare l'eccezione –
    IF(UPPER(mansione_dip) != 'ISTRUTTORE' OR mansione_dip IS NULL) THEN
      RAISE mansione_exc;
    END IF;

    -- Inserimento della nuova scheda aventi i dati dell'istruttore e dell'utente, a cui si vuole associare la scheda, insieme ai dati --
    -- relativi all'inizio della scheda e alla sua scadenza --
    INSERT INTO Scheda_di_allenamento VALUES(data_inizio_schedaX, fk_scheda_utente, fk_dipendente, data_fine_schedaX);

    -- Inserimento del primo esercizio --
    -- La tabella "Coniente" rappresenta i dati degli esercizi contenuti all'interno di una scheda di allenamento --
    INSERT INTO Contiene VALUES(numero_serie, data_inizio_schedaX, fk_scheda_utente, fk_esercizio, peso, numero_ripetizioni,
                                secondi_di_recupero);

    -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
    IF ((numero_serie2 IS NOT NULL) AND (fk_esercizio2 IS NOT NULL) AND (secondi_di_recupero2 IS NOT NULL)) THEN
      INSERT INTO Contiene VALUES(numero_serie2, data_inizio_schedaX, fk_scheda_utente, fk_esercizio2, peso2,
                                  numero_ripetizioni2, secondi_di_recupero2);
    END IF;

    -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
    IF ((numero_serie3 IS NOT NULL) AND (fk_esercizio3 IS NOT NULL) AND (secondi_di_recupero3 IS NOT NULL)) THEN
      INSERT INTO Contiene VALUES(numero_serie3, data_inizio_schedaX, fk_scheda_utente, fk_esercizio3, peso3,
                                  numero_ripetizioni3, secondi_di_recupero3);
    END IF;

    -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
    IF ((numero_serie4 IS NOT NULL) AND (fk_esercizio4 IS NOT NULL) AND (secondi_di_recupero4 IS NOT NULL)) THEN
      INSERT INTO Contiene VALUES(numero_serie4, data_inizio_schedaX, fk_scheda_utente, fk_esercizio4, peso4,
                                  numero_ripetizioni4, secondi_di_recupero4);
    END IF;

    -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
    IF ((numero_serie5 IS NOT NULL) AND (fk_esercizio5 IS NOT NULL) AND (secondi_di_recupero5 IS NOT NULL)) THEN
      INSERT INTO Contiene VALUES(numero_serie5, data_inizio_schedaX, fk_scheda_utente, fk_esercizio5, peso5,
                                  numero_ripetizioni5, secondi_di_recupero5);
    END IF;

    -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
    IF ((numero_serie6 IS NOT NULL) AND (fk_esercizio6 IS NOT NULL) AND (secondi_di_recupero6 IS NOT NULL)) THEN
      INSERT INTO Contiene VALUES(numero_serie6, data_inizio_schedaX, fk_scheda_utente, fk_esercizio6, peso6,
                                  numero_ripetizioni6, secondi_di_recupero6);
    END IF;

    -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
    IF ((numero_serie7 IS NOT NULL) AND (fk_esercizio7 IS NOT NULL) AND (secondi_di_recupero7 IS NOT NULL)) THEN
      INSERT INTO Contiene VALUES(numero_serie7, data_inizio_schedaX, fk_scheda_utente, fk_esercizio7, peso7,
```

```
                                    numero_ripetizioni7, secondi_di_recupero7);
  END IF;

  -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
  IF ((numero_serie8 IS NOT NULL) AND (fk_esercizio8 IS NOT NULL) AND (secondi_di_recupero8 IS NOT NULL)) THEN
    INSERT INTO Contiene VALUES(numero_serie8, data_inizio_schedaX, fk_scheda_utente, fk_esercizio8, peso8,
                                    numero_ripetizioni8, secondi_di_recupero8);
  END IF;

  -- Controlliamo se i valori obbligatori non sono null inserendo poi l'esercizio --
  IF ((numero_serie9 IS NOT NULL) AND (fk_esercizio9 IS NOT NULL) AND (secondi_di_recupero9 IS NOT NULL)) THEN
    INSERT INTO Contiene VALUES(numero_serie9, data_inizio_schedaX, fk_scheda_utente, fk_esercizio9, peso9,
                                    numero_ripetizioni9, secondi_di_recupero9);
  END IF;

EXCEPTION
  WHEN mansione_exc THEN
    RAISE_APPLICATION_ERROR('-20001', 'LA MANSIONE DEL DIPENDENTE È DIVERSA DA ISTRUTTORE: ' || mansione_dip);
END;
```

Creazione abbonamento

Questa procedura serve per creare delle nuove tipologie abbonamento con dei corsi compresi. Se il corso non esiste, esso verrà inserito come nuovo corso, insieme alla tipologia abbonamento e alle informazioni contenute nella tabella Compreso. Si possono mettere un massimo di 4 corsi, dove il primo è obbligatorio.

```
--CREAZIONE TIPOLOGIA ABBONAMENTO--

CREATE OR REPLACE PROCEDURE creazione_abbonamento(nome_tipologia_abbonamento VARCHAR2,
                                                  durata_abbonamento NUMBER, costo_abbonamento NUMBER,
                                                   nome_corso1 VARCHAR2, obiettivo1 VARCHAR2 := NULL,
                                                   nome_corso2 VARCHAR2 := NULL, obiettivo2 VARCHAR2 := NULL,
                                                   nome_corso3 VARCHAR2 := NULL, obiettivo3 VARCHAR2 := NULL,
                                                   nome_corso4 VARCHAR2 := NULL, obiettivo4 VARCHAR2 := NULL)
  AS
  BEGIN
    -- Inserimento di una nuova tipologia abbonamento nella tabella "Tipologia_abbonamento" --
    INSERT INTO Tipologia_abbonamento VALUES (nome_tipologia_abbonamento, durata_abbonamento, costo_abbonamento);

    -- Si controlla che il primo corso da inserire non esista all'interno della tabella "Corso" --
    -- Se esiste, esso non verrà inserito nella tabella. Se non esiste, verrà inserito un nuovo corso
    INSERT INTO corso (nome_corso, obiettivo)
      SELECT nome_corso1, obiettivo1
      FROM dual
      WHERE NOT EXISTS(SELECT * FROM corso WHERE nome_corso = nome_corso1);

    --Insertimento dell'associazione tra "Corso" e "Tipologia_abbonamento"--
    INSERT INTO Compreso VALUES (nome_corso1, nome_tipologia_abbonamento, durata_abbonamento);

    --Controllo per l'inserimento di nuovi corsi--
    IF (nome_corso2 IS NOT NULL) THEN
      --si controlla se il corso non esiste e nel caso lo si crea--
      INSERT INTO corso (nome_corso, obiettivo)
        SELECT nome_corso2, obiettivo2
        FROM dual
        WHERE NOT EXISTS(SELECT * FROM corso WHERE nome_corso = nome_corso2);

      --si associa il corso alla tipologia abbonamento--
      INSERT INTO Compreso VALUES (nome_corso2, nome_tipologia_abbonamento, durata_abbonamento);
    END IF;

    --si controlla se si vuole aggiunge un ulteriore corso all'abbonamento--
    IF (nome_corso3 IS NOT NULL) THEN
```

```
        --si controlla se il corso non esiste e nel caso lo si crea--
        INSERT INTO corso (nome_corso, obiettivo)
          SELECT nome_corso3, obiettivo3
          FROM dual
          WHERE NOT EXISTS(SELECT * FROM corso WHERE nome_corso = nome_corso3);

        --si associa il corso alla tipologia abbonamento--
        INSERT INTO Compreso VALUES (nome_corso3, nome_tipologia_abbonamento, durata_abbonamento);
      END IF;

      --si controlla se si vuole aggiunge un ulteriore corso all'abbonamento--
      IF (nome_corso4 IS NOT NULL) THEN
        --si controlla se il corso non esiste e nel caso lo si crea--
        INSERT INTO corso (nome_corso, obiettivo)
          SELECT nome_corso4, obiettivo4
          FROM dual
          WHERE NOT EXISTS(SELECT * FROM corso WHERE nome_corso = nome_corso4);

        --si associa il corso alla tipologia abbonamento--
        INSERT INTO Compreso VALUES (nome_corso4, nome_tipologia_abbonamento, durata_abbonamento);
      END IF;
    END;
```

Genera utenza

Questa procedura permette la creazione di utenze casuali. Utilizzata per lo scheduler che verrà elencato più avanti nei paragrafi.

```
--GENERA UTENZE--
create or replace PROCEDURE genera_utenza
AS
    importo Utenza.importo_utenza%type;
    numero_fat Utenza.numero_fattura%type;
    data_scad DATE := CURRENT_DATE + INTERVAL '7' DAY;

    CURSOR C IS (SELECT * FROM Sede);
BEGIN
    FOR s IN C
    LOOP
      IF (s.stato_sede = 'ATTIVO') THEN
        importo := trunc(dbms_random.value(1, 999.99),2);
        numero_fat := dbms_random.string('X', 5);

        INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza,
                           tipologia_utenza)
        VALUES(numero_fat, s.via_sede, s.civico_sede, s.cap_sede, importo, data_scad, 'ENEL-LUCE');

        importo := trunc(dbms_random.value(1, 999.99),2);
        numero_fat := dbms_random.string('X', 5);
        INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza,
                           tipologia_utenza)
        VALUES(numero_fat, s.via_sede, s.civico_sede, s.cap_sede, importo, data_scad, 'ENEL-ELETTRICITA');

        importo := trunc(dbms_random.value(1, 999.99),2);
        numero_fat := dbms_random.string('X', 5);
        INSERT INTO Utenza(numero_fattura, fk_sede_via, fk_sede_civico, fk_sede_cap, importo_utenza, data_scadenza,
                           tipologia_utenza)
        VALUES(numero_fat, s.via_sede, s.civico_sede, s.cap_sede, importo, data_scad, 'ACQUA');
      END IF;
    END LOOP;
END;
```

Licenziamento dipendente

Questa procedura ha lo scopo di licenziare un dipendente. Il licenziamento non è testimoniato dall'assenza del dipendente, ma dall'aggiornamento dei suoi dati, in particolare con l'aggiunta della data di licenziamento.

```
--LICENZIAMENTO--
CREATE OR REPLACE PROCEDURE licenziamento_dipendente(numero_documento CHAR)
AS
BEGIN
   -- Aggiorniamo la tabella "Dipendente" con il dato relativo alla data in cui è stato licenziato --
   UPDATE Dipendente SET data_di_licenziamento = SYSDATE WHERE Dipendente.fk_persona = numero_documento;
END;
```

Pagamento utenze

Questa procedura serve per pagare tutte le utenze di una sede specifica.

```
--PAGAMENTO UTENZA--
CREATE OR REPLACE PROCEDURE pagamento_utenze(via_sede VARCHAR2, civico_sede VARCHAR2, cap_sede NUMBER)
AS
BEGIN
   -- Aggiornamento di "pagamento_utenza" con il valore "S", per identificare che le utenze sono state pagate, di una specifica sede
   UPDATE Utenza SET pagamento_utenza = 'S'
      WHERE fk_sede_via = via_sede AND fk_sede_civico = civico_sede AND fk_sede_cap = cap_sede;
END;
```

Prenotazione piscina

Questa procedura serve a compilare i dati relativi ad una prenotazione effettuata da parte dell'utente.

```
--PRENOTAZIONE PISCINA--

CREATE OR REPLACE PROCEDURE prenotazione_piscina(numero_utente CHAR, via_piscina VARCHAR2, civico_piscina VARCHAR2,
                                                 cap_piscina NUMBER, numero_piscina NUMBER, data_prenotazione VARCHAR2)
AS
   -- ECCEZIONI --
   -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO" --
   sede_exc EXCEPTION;
   -- Questa eccezione viene lanciata nel caso in cui l'orario della prenotazione sia inferiore delle 7 e superiore alle 23 --
   prenotazione_exc EXCEPTION;

   -- VARIABILI --
   stato_sedeX Sede.stato_sede%TYPE;
   data_prenotazioneX TIMESTAMP := to_timestamp(data_prenotazione, 'DD-MM-YY HH24:MI');
   ora_data_prenotazione NUMBER;

BEGIN
   ora_data_prenotazione := to_number(to_char(data_prenotazioneX, 'HH24.MI'));

   -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
   SELECT stato_sede INTO stato_sedeX FROM Sede
   WHERE via_sede = via_piscina AND civico_sede = civico_piscina AND cap_sede = cap_piscina;

   IF(stato_sedeX = 'NON ATTIVO') THEN
      RAISE sede_exc;
   END IF;

   -- Controlliamo se l'orario della prenotazione non sia compreso tra l'ora di apertura delle sedi e l'orario di chiusura --
   IF(ora_data_prenotazione NOT BETWEEN 7.00 AND 23.00) THEN
      RAISE prenotazione_exc;
   END IF;
```

```
   -- Inseriamo i dati relativi alla prenotazione dell'utente iscritto --
   INSERT INTO Prenotazione VALUES(data_prenotazioneX, numero_utente, numero_piscina, via_piscina, civico_piscina,
                                   cap_piscina, 20.00);

EXCEPTION
   WHEN sede_exc THEN
      RAISE_APPLICATION_ERROR('-20007', 'LA SEDE NON È ATTIVA');

   WHEN prenotazione_exc THEN
      RAISE_APPLICATION_ERROR('-20010', 'NON È CONSENTITA LA PRENOTAZIONE: 7:00 > ' || ora_data_prenotazione || ' <
23:00');
END;
```

Registra turno

Questa procedura è utile al dipendente per effettuare la propria presenza. Questa procedura ha un duplice scopo: registrare la presenza in entrata ed in uscita. La procedura controlla se l'utente ha già effettuato una presenza durante il proprio turno di lavoro. Se non l'avesse effettuata, allora registra l'entrata, altrimenti l'uscita.

```
--REGISTRA TURNO--

CREATE OR REPLACE PROCEDURE registra_turno(numero_dipendente VARCHAR2)
AS
      -- ECCEZIONI --
      -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO" --
      sede_exc EXCEPTION;
      -- Questa eccezione viene lanciata nel caso in cui il dipendente che vuole effettuare la presenza risulta lincenziato
      licenziato_exc EXCEPTION;

      -- CURSORI --
      -- Questo cursore seleziona tutte le assegnazione della data odierna del dipendente e le relative informazioni associate
ottenute dalla tabella "Tipologia_turno" --

       CURSOR C IS (select * from Assegnazione_settimanale join Tipologia_turno
                                 on fk_tipologia_turno_nome = nome_tipologia_turno
                                 where fk_dipendente = numero_dipendente
                                 and data_assegnazione = to_date(sysdate, 'DD-MM-YY'));

      -- VARIABILI --
      stato_sedeX Sede.stato_sede%TYPE;
      ora_inizio_turnoX NUMBER := to_number(to_char(current_timestamp, 'HH24.MI'));
      dip Dipendente%rowtype;
      count_presenze_dipendente NUMBER;

BEGIN
      -- Prendiamo le informazioni relativi al dipendente --
      SELECT * INTO dip FROM Dipendente WHERE fk_persona = numero_dipendente;

      -- Controlliamo che l'utente non sia stato licenziato in precedenza --
      IF (dip.data_di_licenziamento IS NOT NULL) THEN
         RAISE licenziato_exc;
      END IF;

      -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
      SELECT stato_sede INTO stato_sedeX FROM Sede
         WHERE via_sede = dip.fk_sede_via AND civico_sede = dip.fk_sede_civico AND cap_sede = dip.fk_sede_cap;

      IF (stato_sedeX = 'NON ATTIVO') THEN
         RAISE sede_exc;
      END IF;
```

```
    -- Viene effettuato un ciclo per ogni assegnazione del dipendente nella data odierna --
    -- Si conta, prima di tutto, quante presenze ha effettuato il dipendente, tenendo conto della data
    -- odierna, del turno a lui assegnato e della data di uscita --
    -- Se il count risultasse 0, vuol dire che non esiste una presenza in entrata effettuata dal dipendente --
    -- nella data odierna durante il suo turno. Per cui, controlliamo se "ora_inizio_turnoX", che rappresenta --
    -- l'orario in cui entra il dipendente, sia compreso tra l'inizio e la fine del suo turno. Se risultasse vero --
    -- allora inseriamo la presenza del dipendente –
    -- Se il count risultasse 1, vuol dire che esiste una presenza del dipendente nella data odierna durante il suo turno --
    -- In questo caso, deduciamo che il dipendente voglia registrare la sua uscita --
    FOR turno_dip IN C
    LOOP
      SELECT count(fk_dipendente) INTO count_presenze_dipendente FROM Presenza_dipendente
        WHERE fk_dipendente = numero_dipendente AND trunc(data_ora_entrata)= to_date(sysdate, 'DD-MM-YY')
        AND UPPER(fk_tipologia_turno_nome) = UPPER(turno_dip.fk_tipologia_turno_nome)
        AND data_ora_uscita IS NULL AND fk_sede_via = dip.fk_sede_via
        AND fk_sede_civico = dip.fk_sede_civico AND fk_sede_cap = dip.fk_sede_cap;

      IF(ora_inizio_turnoX BETWEEN turno_dip.ora_inizio_turno AND turno_dip.ora_fine_turno
         AND count_presenze_dipendente = 0) THEN
           INSERT INTO Presenza_dipendente(data_ora_entrata, fk_dipendente, fk_sede_via, fk_sede_civico, fk_sede_cap,
                                           fk_tipologia_turno_nome)
          VALUES (current_timestamp, numero_dipendente, dip.fk_sede_via, dip.fk_sede_civico, dip.fk_sede_cap,
                  turno_dip.fk_tipologia_turno_nome);
      END IF;

      IF(count_presenze_dipendente = 1) THEN
        UPDATE Presenza_dipendente SET data_ora_uscita = current_timestamp WHERE fk_dipendente = numero_dipendente
          AND UPPER(fk_tipologia_turno_nome) = UPPER(turno_dip.fk_tipologia_turno_nome) AND data_ora_uscita IS NULL;
      END IF;
    END LOOP;

EXCEPTION
    WHEN sede_exc THEN
      RAISE_APPLICATION_ERROR('-20008', 'LA SEDE ' || dip.fk_sede_via || ', '|| dip.fk_sede_civico || ', '|| dip.fk_sede_cap || '
NON È ATTIVA');
    WHEN licenziato_exc THEN
      RAISE_APPLICATION_ERROR('-20009', 'IL DIPENDENTE È STATO LICENZIATO!');
END;
```

Rinnovo sottoscrizione

Questa procedura permette di rinnovare una sottoscrizione di un utente. È possibile inserire solo il numero utente, nel caso in cui si voglia rinnovare la sottoscrizione mantenendo i dati della sede di appartenenza; oppure è possibile rinnovare anche la sede di appartenenza. Questa procedura permette anche di consegnare il certificato medico, senza alterare i dati della sottoscrizione, come la data della scadenza della sottoscrizione.

```
CREATE OR REPLACE PROCEDURE rinnovo_sottoscrizione(numero_utente CHAR,  nome_abbonamento VARCHAR2,
                                                   via_sede VARCHAR2 := NULL,  civico_sede VARCHAR2 := NULL,
                                                   cap_sede NUMBER := NULL)
AS
  -- ECCEZIONI --
  -- Questa eccezione viene lanciata nel caso in cui si sta cercando di rinnovare una sottoscrizione che non ha bisogno di un rinnovo
  pagamento_exc EXCEPTION;
  -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO" --
  sede_exc EXCEPTION;

  -- VARIABILI --
  sottoscrizione_pagata Sottoscrizione.ha_pagato%TYPE;
  certificato_consegnato Sottoscrizione.certificato_medico%TYPE;
  stato_sedeX Sede.stato_sede%type;
```

```
   BEGIN
      -- Reperiamo le informazioni relativi al pagamento ed alla consegna del certificato medico della sottoscrizione da parte
dell'utente --
      -- Se fossero "S" vuol dire che la sottoscrizione risulta pagata, per cui questa sottoscrizione non può essere rinnovata --
      SELECT ha_pagato, certificato_medico INTO sottoscrizione_pagata, certificato_consegnato FROM Sottoscrizione
         WHERE fk_utente = numero_utente AND fk_abbonamento_nome = nome_abbonamento;

   IF(sottoscrizione_pagata = 'S' AND certificato_consegnato = 'S') THEN
         RAISE pagamento_exc;
    END IF;

     -- Se i dati relativi all'indirizzo della sede fossero NULL, significherebbe che si sta cercando di rinnovare la sottoscrizione --
      -- nella sede da cui è stata inserita la prima volta. Per cui, in questo caso, verrà rinnovata la sottoscrizione, senza aggiornare
      -- i dati reletavi a dove è avvenuto il rinnovo --
      IF(via_sede IS NULL AND civico_sede IS NULL AND cap_sede IS NULL) THEN
         UPDATE Sottoscrizione SET ha_pagato = 'S', certificato_medico = 'S'
            WHERE fk_utente = numero_utente AND fk_abbonamento_nome = nome_abbonamento;
      ELSE
         -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
         SELECT stato_sede INTO stato_sedeX FROM Sede WHERE Sede.via_sede = via_sede AND Sede.civico_sede = civico_sede AND
Sede.cap_sede = cap_sede;

         IF (stato_sedeX = 'NON ATTIVO') THEN
            RAISE sede_exc;
         END IF;

      -- Dopo aver controllato che la sede è attiva, il rinnovo verrà effettuato tenendo conto della nuova sede --
         -- da cui è avvenuta la richiesta --
         UPDATE Sottoscrizione SET ha_pagato = 'S', certificato_medico = 'S', fk_sede_via = via_sede,
                           fk_sede_civico = civico_sede, fk_sede_cap = cap_sede
         WHERE fk_utente = numero_utente AND fk_abbonamento_nome = nome_abbonamento;
      END IF;

   EXCEPTION
      WHEN pagamento_exc THEN
         RAISE_APPLICATION_ERROR('-20005', 'LA SOTTOSCRIZIONE ( ' || nome_abbonamento || ' ) DI ' || numero_utente || ' È IN
REGOLA');

      WHEN sede_exc THEN
         RAISE_APPLICATION_ERROR('-20006', 'LA SEDE NON È ATTIVA');
END;
```

Sottoscrizione utente

Questa procedura permette la sottoscrizione di un utente, con l'inserimento dei suoi data anagrafici. Se l'utente non è sottoscritto, questa procedura inserirà i suoi dati e la sottoscrizione alla tipologia di abbonamento scelta. Altrimenti, creerà una nuova sottoscrizione dell'utente che ha già una sottoscrizione attiva o non attiva. Ovviamente verrà impedita l'iscrizione ad un abbonamento che l'utente già possiede.

```
CREATE OR REPLACE PROCEDURE sottoscrizione_utente (numero_persona   CHAR, nome_abbonamento VARCHAR2,
                                                   via_sedeX       VARCHAR2,  civico_sedeX     VARCHAR2,
                                                    cap_sedeX       NUMBER,   codice_fiscale   CHAR := NULL,
                                                     data_di_nascita  VARCHAR2 := NULL,  nome          VARCHAR2 := NULL,
                                                    cognome        VARCHAR2 := NULL, genere         CHAR := NULL,
                                                     via_persona     VARCHAR2 := NULL,   civico_persona   VARCHAR2 :=
NULL,
                                                     cap_persona     NUMBER := NULL,   telefono_persona CHAR := NULL)
  AS
          -- ECCEZIONI --
          -- Questa eccezione viene lanciata nel caso in cui si voglia sottoscrivere l'utente ad una tipologia abbonamento a cui è già
iscritto --
     sottoscrizione_exc EXCEPTION;
          -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO" --
          sede_exc EXCEPTION;

          -- VARIABILI --
     data_di_nascitaX DATE := TO_DATE ( data_di_nascita, 'DD-MM-YY' );
     eta_persona NUMBER := TO_NUMBER ( months_between(trunc(sysdate), data_di_nascitaX) / 12 );
     check_sot NUMBER := 0;
     stato_sedeX Sede.stato_sede%type;
  BEGIN
          -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
     SELECT stato_sede INTO stato_sedeX FROM Sede
       WHERE via_sede = via_sedeX AND civico_sede = civico_sedeX AND cap_sede = cap_sedeX;

     IF (stato_sedeX = 'NON ATTIVO') THEN
       RAISE sede_exc;
     END IF;

     -- Controlliamo se il codice fiscale in input e la data di nascita siano NULL --
     -- Nel caso in cui non fossero NULL, vuol dire che si sta cercando di sottoscrivere una persona per la prima volta ad un
abbonamento --
    -- In quanto in caso contrario, vuol dire che la persona è già presente e si vuole sottoscrivere ad un nuovo abbonamento --
     IF (codice_fiscale IS NOT NULL AND data_di_nascita IS NOT NULL) THEN
       -- Inseriamo i dati relativi alla persona --
       INSERT INTO persona
         VALUES (numero_persona,  nome,  cognome,  genere,  codice_fiscale,  data_di_nascitaX,  eta_persona,  via_persona,
           civico_persona, cap_persona,  telefono_persona );

          -- Inseriamo l'utente associato alla persona appena creata --
          INSERT INTO Utente(fk_persona) VALUES ( numero_persona );
     END IF;

     -- Tramite un contatore controlliamo se esiste già una sottoscrizione dell'utente allo specifico abbonamento --
     SELECT COUNT(id_sottoscrizione) INTO check_sot FROM sottoscrizione
     WHERE fk_utente = numero_persona AND fk_abbonamento_nome = nome_abbonamento;
     -- Se esiste avremo il contatore maggiore di 0 e mandiamo l'errore --
     IF ( check_sot > 0 ) THEN
       RAISE sottoscrizione_exc;
     ELSE
       -- Inseriamo la sottoscrizione --
       INSERT INTO sottoscrizione (fk_utente,  fk_abbonamento_nome,  fk_sede_via,  fk_sede_civico,  fk_sede_cap)
```

```
      VALUES (   numero_persona,  nome_abbonamento,  via_sedeX,  civico_sedeX,  cap_sedeX);
    END IF;

  EXCEPTION
    WHEN sottoscrizione_exc THEN
      raise_application_error('-20003', 'L''UTENTE È GIÀ SOTTOSCRITTO A:  ' || nome_abbonamento);

    WHEN sede_exc THEN
      raise_application_error('-20004', 'LA SEDE ' || via_sedeX || ', '|| civico_sedeX || ', '|| cap_sedeX || ' NON È ATTIVA');
  END;
```

Vendita prodotto

Questa procedura permette la registrazione della vendita di uno o più prodotti (max 5). Per far sì che la procedura funzioni, è obbligatorio l'inserimento del primo prodotto.

```
--VENDITA PRODOTTI--

create or replace PROCEDURE vendita_prodotti(sede_via VARCHAR2, sede_civico VARCHAR2, sede_cap VARCHAR2,
                                             codice_a_barre1 CHAR,  quantita_venduta1 NUMBER,
                                             codice_a_barre2 CHAR := NULL, quantita_venduta2 NUMBER := NULL,
                                             codice_a_barre3 CHAR := NULL, quantita_vendut3 NUMBER := NULL,
                                             codice_a_barre4 CHAR := NULL, quantita_venduta4 NUMBER := NULL,
                                             codice_a_barre5 CHAR := NULL, quantita_venduta5 NUMBER := NULL)
  AS
          -- ECCEZIONI --
          -- Questa eccezione viene lanciata nel caso in cui la sede inserita si trova nello stato "NON ATTIVO"
          sede_exc EXCEPTION;

          -- Variabili --
          prezzo_vendita_un Prodotto.prezzo_vendita_unitario%type;
          prezzo_totX Scontrino.prezzo_totale%type := 0;
          stato_sedeX Sede.stato_sede%type;

  BEGIN
    -- Controlliamo se la sede è nello stato "ATTIVO" o "NON ATTIVO" --
    SELECT stato_sede INTO stato_sedeX FROM Sede
      WHERE via_sede = sede_via AND civico_sede = sede_civico AND cap_sede = sede_cap;

    IF (stato_sedeX = 'NON ATTIVO') THEN
      RAISE sede_exc;
    END IF;

    -- Reperiamo il prezzo di vendita unitario del primo prodotto selezionato dalla tabella "Prodotto" --
    SELECT prezzo_vendita_unitario INTO prezzo_vendita_un
      FROM Prodotto where codice_a_barre = codice_a_barre1;

    -- Calcolo del totale dello scontrino --
    prezzo_totX := prezzo_totX + (prezzo_vendita_un * quantita_venduta1);

    -- Controlliamo se è stato inserito un ulteriore prodotto da vendere --
    IF (codice_a_barre2 IS NOT NULL AND quantita_venduta2 IS NOT NULL) THEN
          -- Reperiamo il prezzo di vendita unitario del primo prodotto selezionato dalla tabella "Prodotto" --
      SELECT prezzo_vendita_unitario INTO prezzo_vendita_un
        FROM Prodotto where codice_a_barre = codice_a_barre2;

          -- Calcolo del totale dello scontrino --
      prezzo_totX := prezzo_totX + (prezzo_vendita_un * quantita_venduta2);
    END IF;

   -- Controlliamo se è stato inserito un ulteriore prodotto da vendere –
    IF (codice_a_barre3 IS NOT NULL AND quantita_venduta3 IS NOT NULL) THEN
          -- Reperiamo il prezzo di vendita unitario del primo prodotto selezionato dalla tabella "Prodotto" --
```

```
    SELECT prezzo_vendita_unitario INTO prezzo_vendita_un
      FROM Prodotto where codice_a_barre = codice_a_barre3;

       -- Calcolo del totale dello scontrino --
    prezzo_totX := prezzo_totX + (prezzo_vendita_un * quantita_venduta3);
  END IF;

 -- Controlliamo se è stato inserito un ulteriore prodotto da vendere --
  IF (codice_a_barre4 IS NOT NULL AND quantita_venduta4 IS NOT NULL) THEN
       -- Reperiamo il prezzo di vendita unitario del primo prodotto selezionato dalla tabella "Prodotto" --
    SELECT prezzo_vendita_unitario INTO prezzo_vendita_un
      FROM Prodotto where codice_a_barre = codice_a_barre4;

       -- Calcolo del totale dello scontrino --
    prezzo_totX := prezzo_totX + (prezzo_vendita_un * quantita_venduta4);
  END IF;

 -- Controlliamo se è stato inserito un ulteriore prodotto da vendere --
  IF (codice_a_barre5 IS NOT NULL AND quantita_venduta5 IS NOT NULL) THEN
       -- Reperiamo il prezzo di vendita unitario del primo prodotto selezionato dalla tabella "Prodotto" --
    SELECT prezzo_vendita_unitario INTO prezzo_vendita_un
      FROM Prodotto where codice_a_barre = codice_a_barre5;

       -- Calcolo del totale dello scontrino --
    prezzo_totX := prezzo_totX + (prezzo_vendita_un * quantita_venduta5);
  END IF;

  -- Creazione dello scontrino --
  INSERT INTO Scontrino (fk_sede_via, fk_sede_civico, fk_sede_cap, prezzo_totale)
    VALUES (sede_via, sede_civico, sede_cap, prezzo_totX);

  -- Inseriamo i dati relativi alla vendita del/dei prodotto/i, associando lo scontrino, il/i prodotto/i e la quantità venduta --
  INSERT INTO Vendita VALUES(id_scontrino_seq.currval, codice_a_barre1, quantita_venduta1);

  IF (codice_a_barre2 IS NOT NULL) THEN
    INSERT INTO Vendita VALUES(id_scontrino_seq.currval, codice_a_barre2, quantita_venduta2);
  END IF;

  IF (codice_a_barre3 IS NOT NULL) THEN
    INSERT INTO Vendita VALUES(id_scontrino_seq.currval, codice_a_barre3, quantita_venduta3);
  END IF;

  IF (codice_a_barre4 IS NOT NULL) THEN
    INSERT INTO Vendita VALUES(id_scontrino_seq.currval, codice_a_barre4, quantita_venduta4);
  END IF;

  IF (codice_a_barre5 IS NOT NULL) THEN
    INSERT INTO Vendita VALUES(id_scontrino_seq.currval, codice_a_barre5, quantita_venduta5);
  END IF;

EXCEPTION
  WHEN sede_exc THEN
    RAISE_APPLICATION_ERROR('-20002', 'LA SEDE ' || sede_via || ', '|| sede_civico || ', '|| sede_cap || ' NON È ATTIVA');
END;
```

Verifica entrata

Questa procedura verifica se l'utente può entrare oppure non può entrare all'interno della palestra in base ai dati della sua sottoscrizione. Gli sarà negato l'accesso solo se tutte le sue sottoscrizioni non sono attive.

```
--VERIFICA ENTRATA--
CREATE OR REPLACE PROCEDURE verifica_entrata(numero_tessera_utente CHAR) AS
  -- CURSORE --
  -- Questo cursore seleziona tutti i pagamenti effettuati, e le relative scadenze, --
  -- di ogni sottoscrizione di uno specifico utente dato il numero tessera --
  CURSOR C IS
    SELECT ha_pagato AS pagamento_cur, data_fine_sottoscrizione AS scadenza_cur
    FROM Sottoscrizione JOIN UTENTE ON fk_utente = fk_persona
    WHERE numero_tessera = numero_tessera_utente;

  -- VARIABILI
  info_utente C%rowtype;
  counter_sottoscrizioni_scadute NUMBER := 0;
  tot_sottoscrizioni NUMBER := 0;
  numero_documento Utente.fk_persona%type;

BEGIN
  -- Ricaviamo il numero documento della persona per poter controllare successivamente le sue sottoscrizioni --
  SELECT fk_persona INTO numero_documento FROM Utente WHERE numero_tessera = numero_tessera_utente;

  -- Contiamo quante sono tutte le sottoscrizioni dell'utente --
  SELECT COUNT(id_sottoscrizione) INTO tot_sottoscrizioni FROM Sottoscrizione
    WHERE fk_utente = numero_documento;

  -- Per ogni sottoscrizione contenuta nel cursore, effettuiamo un controllo sulla sottoscrizione --
  -- Se la sottoscrizione non fosse stata pagata o risultasse scaduta, allora aumentiamo un counter --
  -- che simboleggia le sottoscrizioni scadute dell'utente --
  FOR info_utente IN C
  LOOP
    IF(info_utente.pagamento_cur = 'N' OR info_utente.scadenza_cur <= SYSDATE) THEN
      counter_sottoscrizioni_scadute := counter_sottoscrizioni_scadute + 1;
    END IF;
  END LOOP;

  -- Se le sottoscrizioni scadute dell'utente fossero minori di tutti le sue sottoscrizioni, significa che --
  -- esiste qualche sottoscrizione in regola, per cui l'utente può entrare. Altrimenti, verrà notificato che –
  -- l'utente in questione non può entrare --
  IF(counter_sottoscrizioni_scadute < tot_sottoscrizioni) THEN
    dbms_output.put_line('L''utente può entrare');
  ELSE
    dbms_output.put_line('L''utente non può entrare');
  END IF;
END;
```

7.5 – Viste

Le procedure sono la parte più strettamente legata alla logica di business e all'automazione della Base di Dati. Se nel DDL e nel DML il problema principale è decidere la rappresentazione ottima dei dati e garantire rispetto dei vincoli di integrità su di essi definiti, in questa fase il problema è come sfruttare al meglio tale rappresentazione per automatizzarne la gestione e trarne il massimo profitto.

View sottoscrizioni attive

Questa vista visualizza tutte le sottoscrizioni attive ordinate per sottoscrizione

```
-- VISTA PER VISUALIZZARE TUTTE LE SOTTOSCRIZIONI ATTIVE --

CREATE OR REPLACE VIEW view_sottoscrizioni_attive AS
  SELECT
    id_sottoscrizione,
    numero_documento_persona,
    codice_fiscale_persona          AS codice_fiscale,
    nome_persona                    AS nome,
    cognome_persona                 AS cognome,
    numero_tessera,
    fk_abbonamento_nome             AS tipologia_abbonamento,
    data_inizio_sottoscrizione,
    data_fine_sottoscrizione,
    certificato_medico
  FROM Sottoscrizione
    JOIN persona ON fk_utente = numero_documento_persona
            AND ha_pagato = 'S'
            AND data_fine_sottoscrizione > sysdate
    JOIN utente ON fk_persona = fk_utente
  ORDER BY
    id_sottoscrizione;
```

View sottoscrizioni scadute

Questa vista visualizza tutte le sottoscrizioni scadute ordinate per sottoscrizione

```
-- VISTA PER VISUALIZZARE TUTTE LE SOTTOSCRIZIONI SCADUTE --

CREATE OR REPLACE VIEW view_sottoscrizioni_scadute AS
  SELECT
    id_sottoscrizione,
    numero_documento_persona,
    codice_fiscale_persona          AS codice_fiscale,
    nome_persona                    AS nome,
    cognome_persona                 AS cognome,
    numero_tessera,
    fk_abbonamento_nome             AS tipologia_abbonamento,
    data_inizio_sottoscrizione,
    data_fine_sottoscrizione,
    certificato_medico
  FROM Sottoscrizione
    JOIN persona ON fk_utente = numero_documento_persona
            AND ha_pagato = 'N'
            AND data_fine_sottoscrizione <= sysdate
    JOIN utente ON fk_persona = fk_utente
  ORDER BY
    id_sottoscrizione;
```

View lezioni settimanali

Questa vista visualizza tutte le lezioni settimanali ordinate per lezione

```
-- VISTA PER VISUALIZZARE TUTTE LE LEZIONI SETTIMANALI --

CREATE OR REPLACE VIEW view_lezioni_settimanali AS
 SELECT
    id_lezione,
    fk_sala_codice              AS codice_sala,
    fk_corso                    AS corso,
    nome_persona                AS nome_istruttore,
    cognome_persona             AS cognome_istruttore,
    giorno_settimana_lezione    AS giorno,
    ora_inizio,
    ora_fine,
    fk_sala_via                 AS via_sede,
    fk_sala_civico              AS civico_sede,
    fk_sala_cap                 AS cap_sede
  FROM Dipendente
    JOIN persona ON fk_persona = numero_documento_persona
    JOIN lezione_corso ON fk_dipendente = fk_persona
  ORDER BY
    id_lezione;
```

View storico utenze pagate

Questa vista visualizza tutte le utenze pagate in ogni sede e ordinate per data di scadenza.

```
-- VISTA PER VISUALIZZARE LO STORICO DELLE UTENZE PAGATE --

CREATE OR REPLACE VIEW view_storico_utenze_pagate AS
  SELECT
    numero_fattura,
    fk_sede_via         AS via_sede,
    fk_sede_civico      AS civico_sede,
    fk_sede_cap         AS cap_sede,
    data_scadenza,
    tipologia_utenza,
    importo_utenza      AS importo
  FROM
    utenza
  WHERE
    pagamento_utenza = 'S'
  ORDER BY
    data_scadenza;
```

View guadagno mensile sottoscrizioni

Questa vista visualizza il guadagno totale di tutte le sedi dalle sottoscrizioni a partire dal mese scorso fino alla data odierna

```
-- VISTA UTILE A VISUALIZZARE IL GUADAGNO DAL MESE PRECEDENTE DALLE SOTTOSCRIZIONI A PARTIRE DALLA DATA ODIERNA --

CREATE OR REPLACE VIEW view_guadagno_mensile_sott AS
  SELECT
    SUM(costo_abbonamento)    AS totale_guadagno,
    fk_sede_via               AS via_sede,
    fk_sede_civico            AS civico_sede,
    fk_sede_cap               AS cap_sede
  FROM Sottoscrizione
    JOIN tipologia_abbonamento ON fk_abbonamento_nome = nome_tipologia_abbonamento
  WHERE
    data_inizio_sottoscrizione >= add_months(sysdate, - 1)
  GROUP BY
    fk_sede_via,
    fk_sede_civico,
    fk_sede_cap;
```

View guadagno mensile prenotazioni

Questa vista visualizza il guadagno totale di tutte le sedi dalle prenotazioni a partire dal mese scorso fino alla data odierna

```
-- VISTA UTILE A VISUALIZZARE IL GUADAGNO DAL MESE PRECEDENTE DALLE PRENOTAZIONI A PARTIRE DALLA DATA ODIERNA --

CREATE OR REPLACE VIEW view_guadagno_prenotazioni AS
  SELECT
    SUM(prezzo_prenotazione)   AS totale_guadagno,
    fk_piscina_via             AS via_sede,
    fk_piscina_civico          AS civico_sede,
    fk_piscina_cap             AS cap_sede
  FROM Prenotazione
  WHERE
    data_ora_prenotazione >= add_months(sysdate, - 1)
  GROUP BY
    fk_piscina_via,
    fk_piscina_civico,
    fk_piscina_cap
```

View guadagno mensile scontrini

Questa vista visualizza il guadagno totale di tutte le sedi dagli scontrini a partire dal mese scorso fino alla data odierna

```
-- VISTA UTILE A VISUALIZZARE IL GUADAGNO DAL MESE PRECEDENTE DAGLI SCONTRINI A PARTIRE DALLA DATA ODIERNA --

CREATE OR REPLACE VIEW view_guadagno_mensile_scon AS
  SELECT
    SUM(prezzo_totale)    AS totale_guadagno,
    fk_sede_via           AS via_sede,
    fk_sede_civico        AS civico_sede,
    fk_sede_cap           AS cap_sede
  FROM Scontrino
  WHERE
    data_scontrino >= add_months(sysdate, - 1)
  GROUP BY
    fk_sede_via,
    fk_sede_civico,
    fk_sede_cap
```

View ritardi effettuati

Questa vista visualizza i dipendenti che hanno effettuato dei ritardi durante il proprio turno di lavoro

```
-- VISTA UTILE A VISUALIZZARE I DIPENDENTI CHE DURANTE IL PROPRIO TURNO SONO ENTRATI IN RITARDO --

CREATE OR REPLACE VIEW view_ritardi_effettuati AS
  SELECT
        fk_dipendente                   AS Dipendente,
        nome_persona                    AS Nome,
        cognome_persona                 AS Cognome,
        data_ora_entrata                AS Entrata_dipendente,
        fk_tipologia_turno_nome         AS Turno_assegnato,
        ora_inizio_turno                AS Ora_turno_entrata,
        Presenza_dipendente.fk_sede_via AS Via,
        Presenza_dipendente.fk_sede_civico AS Civico,
        Presenza_dipendente.fk_sede_cap AS Cap
   FROM Presenza_dipendente
        JOIN Tipologia_turno ON fk_tipologia_turno_nome = nome_tipologia_turno
            JOIN Dipendente ON fk_dipendente = fk_persona
                JOIN Persona ON fk_persona = numero_documento_persona
  WHERE
        to_number(to_char(data_ora_entrata, 'HH24.MI')) >= (ora_inizio_turno + 0.10)
```

View prodotti in esaurimento

Questa vista visualizza i prodotti che sono quasi in esaurimento

```
-- VISTA UTILE A VISUALIZZARE I PRODOTTI CHE SONO IN ESAURIMENTO --

CREATE OR REPLACE VIEW view_prodotti_in_esaurimento AS
    SELECT
            codice_a_barre,
            nome_prodotto,
            giacenza
    FROM Prodotto
    WHERE
            giacenza < 10
```

View corsi sottoscritti

Questa vista visualizza i corsi con più sottoscrizioni

```
-- VISTA UTILE A VISUALIZZARE QUANTE VOLTE UN CORSO È COMPRESO IN UNA SOTTOSCRIZIONE --

CREATE OR REPLACE VIEW view_corsi_sottoscritti AS
  SELECT
     count(fk_corso)    AS TOT_SOTTOSCRIZIONI_CORSO,
     fk_corso           AS CORSO
  FROM
  Sottoscrizione JOIN Tipologia_abbonamento ON Sottoscrizione.fk_abbonamento_nome =
Tipologia_abbonamento.nome_tipologia_abbonamento
    JOIN Compreso ON Tipologia_abbonamento.nome_tipologia_abbonamento = Compreso.fk_abbonamento_nome
            GROUP BY
               fk_corso
            ORDER BY
               TOT_SOTTOSCRIZIONI_CORSO DESC
```

View dipendenti non assegnati

Questa vista visualizza i dipendenti che non sono ancora stati assegnati per un turno di lavoro nella settimana prossima

```
-- VISTA UTILE A VISUALIZZARE TUTTI I DIPENDENTI CHE NON SONO STATI ASSEGNATI NELLA SETTIMANA PROSSIMA --

CREATE OR REPLACE VIEW view_dipendenti_non_assegnati AS
   SELECT DISTINCT
      fk_dipendente             AS Numero_dipendente,
      nome_persona              AS Nome,
      cognome_persona           AS cognome
   FROM
      Assegnazione_settimanale JOIN Dipendente ON Assegnazione_settimanale.fk_dipendente = Dipendente.fk_persona
          JOIN Persona ON Dipendente.fk_persona = Persona.numero_documento_persona
    WHERE
       fk_dipendente NOT IN (SELECT fk_dipendente FROM Assegnazione_settimanale WHERE data_assegnazione BETWEEN TRUNC(SYSDATE +
7,'D') + 1 AND TRUNC(SYSDATE + 7,'D') + 6)
```

7.6 – Data Control Language

```
--  CREAZIONE RUOLI E UTENTI
-- Creazione Amministratore e relativa assegnazione privilegi
CREATE USER Admin_Gymbro IDENTIFIED BY oracle;
GRANT ALL PRIVILEGES TO Admin_Gymbro;

-- Creazione Ruoli
CREATE ROLE Responsabile_sede;
CREATE ROLE Segretaria;
CREATE ROLE Istruttore;
CREATE ROLE Utente;

-- Privilegi di sistema
GRANT CONNECT, CREATE SESSION TO Responsabile_sede;
GRANT CONNECT, CREATE SESSION TO Segretaria;
GRANT CONNECT, CREATE SESSION TO Istruttore;
GRANT CONNECT, CREATE SESSION TO Utente;

-- Privilegi di oggetto per Segretaria
GRANT SELECT, INSERT, UPDATE ON Persona TO Segretaria;
GRANT SELECT, INSERT, UPDATE ON Utente TO Segretaria;
GRANT SELECT, INSERT, UPDATE ON Prenotazione TO Segretaria;
GRANT SELECT, INSERT ON Vendita TO Segretaria;
GRANT SELECT, INSERT, UPDATE ON Sottoscrizione TO Segretaria;
GRANT SELECT ON Lezione_corso TO Segretaria;
GRANT SELECT ON Tipologia_abbonamento TO Segretaria;
GRANT SELECT ON Compreso TO Segretaria;
GRANT SELECT ON Corso TO Segretaria;
GRANT SELECT ON Assegnazione_settimanale TO Segretaria;
GRANT SELECT, INSERT ON Scontrino TO Segretaria;
GRANT SELECT ON VIEW_CORSI_SOTTOSCRITTI TO Segretaria;
GRANT SELECT ON VIEW_LEZIONI_SETTIMANALI TO Segretaria;
GRANT SELECT ON VIEW_SOTTOSCRIZIONI_ATTIVE TO Segretaria;
GRANT SELECT ON VIEW_SOTTOSCRIZIONI_ATTIVE TO Segretaria;
GRANT SELECT ON VIEW_SOTTOSCRIZIONI_SCADUTE TO Segretaria;
GRANT EXECUTE ON prenotazione_piscina TO Segretaria;
GRANT EXECUTE ON registra_turno TO Segretaria;
GRANT EXECUTE ON rinnovo_sottoscrizione TO Segretaria;
GRANT EXECUTE ON sottoscrizione_utente TO Segretaria;
GRANT EXECUTE ON vendita_prodotti TO Segretaria;

-- Privilegi di oggetto per Responsabile_sede
GRANT SELECT, INSERT, UPDATE ON Assegnazione_settimanale TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Dipendente TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Persona TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Esercizio TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Lezione_corso TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Piscina TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Presenza_dipendente TO Responsabile_sede;
GRANT SELECT, INSERT, UPDATE ON Prodotto TO Responsabile_sede;
```

```
GRANT SELECT, INSERT, UPDATE ON Sala TO Responsabile_sede;
GRANT SELECT, INSERT ON Scontrino TO Responsabile_sede;
GRANT SELECT ON Tipologia_abbonamento TO Responsabile_sede;
GRANT SELECT ON Tipologia_turno TO Responsabile_sede;
GRANT SELECT ON Sede TO Responsabile_sede;
GRANT SELECT ON Utenza TO Responsabile_sede;
GRANT SELECT ON Compreso TO Responsabile_sede;
GRANT SELECT ON Corso TO Responsabile_sede;
GRANT SELECT ON Sottoscrizione TO Responsabile_sede;
GRANT SELECT ON Vendita TO Responsabile_sede;

GRANT SELECT ON VIEW_PRODOTTI_IN_ESAURIMENTO TO Responsabile_sede;
GRANT SELECT ON VIEW_LEZIONI_SETTIMANALI TO Responsabile_sede;
GRANT SELECT ON VIEW_RITARDI_EFFETTUATI TO Responsabile_sede;
GRANT SELECT ON VIEW_DIPENDENTI_NON_ASSEGNATI TO Responsabile_sede;
GRANT SELECT ON VIEW_CORSI_SOTTOSCRITTI TO Responsabile_sede;

GRANT EXECUTE ON assegna_dipendenti TO Responsabile_sede;
GRANT EXECUTE ON assunzione TO Responsabile_sede;
GRANT EXECUTE ON licenziamento_dipendente TO Responsabile_sede;

-- Privilegi di oggetto per Istruttore
GRANT SELECT, INSERT, UPDATE ON Scheda_di_allenamento TO Istruttore;
GRANT SELECT, INSERT, UPDATE ON Contiene TO Istruttore;
GRANT SELECT ON Lezione_corso TO Istruttore;
GRANT SELECT ON Assegnazione_settimanale TO Istruttore;
GRANT SELECT ON Esercizio TO Istruttore;

GRANT EXECUTE ON compila_scheda TO Istruttore;
GRANT EXECUTE ON registra_turno TO Istruttore;

-- Privilegi di oggetto per Utente
GRANT EXECUTE ON verifica_entrata TO Utente;

-- Creazione utenti e relativa assegnazione ruoli
CREATE USER seg1 IDENTIFIED BY Segretaria;
CREATE USER seg2 IDENTIFIED BY Segretaria;
CREATE USER seg3 IDENTIFIED BY Segretaria;
CREATE USER resp1 IDENTIFIED BY Responsabile_sede;
CREATE USER resp2 IDENTIFIED BY Responsabile_sede;
CREATE USER resp3 IDENTIFIED BY Responsabile_sede;
CREATE USER istr1 IDENTIFIED BY Istruttore;
CREATE USER istr2 IDENTIFIED BY Istruttore;
CREATE USER istr3 IDENTIFIED BY Istruttore;
CREATE USER ut1 IDENTIFIED BY Utente;
CREATE USER ut2 IDENTIFIED BY Utente;
CREATE USER ut3 IDENTIFIED BY Utente;

GRANT Responsabile_sede TO resp1, resp2, resp3;
GRANT Segretaria TO seg1, seg2, seg3;
GRANT Istruttore TO istr1, istr2, istr3;
GRANT Utente TO ut1, ut2, ut3;
```

7.7 – Scheduler

Lo scheduler serve a programmare azioni, normalmente periodiche, che avvengono in istanti predeterminati. Similmente ad un trigger, vale il paradigma Evento-Condizione-Azione (ECA), ma l'evento è legato allo scorrere del tempo — quindi all'orologio di sistema — piuttosto che ad un'operazione DML. È possibile programmare operazioni periodiche di manutenzione, auditing, backup, pulitura o aggiornamento dei dati; sia di tipo amministrativo che da utente comune.

```
-- JOB CHE OGNI OGNI 4 MESI GENERA LE UTENZE PER OGNI SEDE ATTIVA
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name            =>    'Genera_Utenze_Sedi',
  job_type            =>    'PLSQL_BLOCK',
  job_action          =>    'genera_utenza',
  start_date          =>    TO_DATE('01-01-2024','DD-MM-YYYY'),
  repeat_interval     =>    'FREQ=MONTHLY; INTERVAL=4',
  enabled             =>    TRUE,
  comments            =>    'Genera le utenze per le sedi attive');
END;

-- JOB CHE OGNI OGNI 5 GENNAIO RESETTA IL CERTIFICATO MEDICO
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name            =>    'Reset_Certificato_medico',
  job_type            =>    'PLSQL_BLOCK',
  job_action          =>    'BEGIN
                              UPDATE Sottoscrizione SET certificato_medico = N;
                             END;',
  start_date          =>    TO_DATE('05-01-2024','DD-MM-YYYY'),
  repeat_interval     =>    'FREQ=YEARLY',
  enabled             =>    TRUE,
  comments            =>    'Reset a N del certificato medico');
END;

-- JOB CHE OGNI PRIMO DEL MESE CANCELLA TUTTE LE PRENOTAZIONI PRECEDENTI DI DUE ANNI
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name            =>    'Cancellazione_Prenotazioni',
  job_type            =>    'PLSQL_BLOCK',
  job_action          =>    'BEGIN
                             DELETE FROM prenotazione
                               WHERE data_fine_attivita < SYSDATE-730;
                            END;',
  start_date          =>    TO_DATE('01-01-2024','DD-MM-YYYY'),
  repeat_interval     =>    'FREQ=MONTHLY',
  enabled             =>    TRUE,
  comments            =>    'Cancellazione delle vecchie prenotazioni');
END;

-- JOB CHE OGNI GIORNO CONTROLLA SE C'È QUALCHE SOTTOSCRIZIONE SCADUTA
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name            =>    'Sottoscrizione_scaduta',
  job_type            =>    'PLSQL_BLOCK',
  job_action          =>    'BEGIN
                                UPDATE Sottoscrizione SET ha_pagato = N
                                    WHERE data_fine_sottoscrizione = CURRENT_DATE;
                             END;',
  start_date          =>    TO_DATE('19-06-2024','DD-MM-YYYY'),
  repeat_interval     =>    'FREQ=DAILY',
  enabled             =>    TRUE,
  comments            =>    'Set a N ha_pagato delle sottoscrizioni scadute oggi');
END;
```