UNIVERSITA' DEGLI STUDI DI NAPOLI PARTHENOPE

DIPARTIMENTO DI SCIENZE E TECNOLOGIE

CORSO DI BASI DI DATI E LABORATORIO DI BASI DI DATI

Parthenope Mental Healt.

Manicomio psichiatrico per fuoricorso.

Ideato e Progettato da:

Ariano Luigi: Mat. 0124002483

Biondi Morgan: Mat. 0124002876

Tridente Antonio: Mat. 0124002692

Anno Accademico:

2022/2023

Indice:

Introduzione

Parthenope Mental Healt, è il nome del manicomio più temuto d'Italia, nel quale vengono rinchiusi tutti i fuoricorso di ogni università italiana. Scherzi a parte, il database qui progettato si occupa di definire la gestione di un manicomio, non solo riguardo al trattamento dei vari pazienti, ma anche riguardo a chi lavora attivamente al suo interno, come medici, infermieri, custodi, e di come essi svolgano le loro mansioni all'interno della struttura divisa in reparti.

Nonostante ciò, particolare attenzione viene comunque fatta al paziente e ai trattamenti che ad esso vengono associati, come la somministrazione dei farmaci da parte degli infermieri, la diagnostica della loro condizione/i da parte degli psichiatri, e le cure a loro assegnate come i trattamenti terapeutici svolti dagli psicoterapeuti. I medici si occupano di effettuare controlli di routine ai pazienti, e se questi risultano malati vengono mandati in ricovero.

Dei reparti si memorizzeranno le caratteristiche peculiari, come il numero stanze di un reparto camere, e gli vengono assegnate funzioni specifiche, ad esempio nel reparto psichiatria si tengono le sedute terapeutiche ed in quello infermieristico risultano i ricoveri causati dai controlli negativi, della mensa quanti posti essa metta a disposizione ed il numero di stanze disponibili per il reparto camere

Nella struttura lavorano anche figure di servizio come i segretari che gestiscono le prenotazioni effettuate da eventuali visitatori, prenotazioni che possono essere accettate oppure no, inservienti che giornalmente lavorano in reparti diversi per contribuire ad una corretta igiene e pulizia della struttura ed infine i custodi, ad ognuno dei quali è assegnato un reparto.

Per garantire una maggiore sicurezza nella struttura vi sono presenti anche delle guardie, incaricate di vigilare sui vari reparti ed alle quali vengono assegnate delle armi di servizio come pistole, teaser o peggio...

Detto ciò, nelle pagine successive verranno affrontati diversi capitoli nei quali verranno espresse meglio le fasi di progettazione, passando dal diagramma EE/R alla traduzione all’implementazione in linguaggio SQL, per ognuna delle fasi un corollario per descrivere le caratteristiche più centrali come le entità e le associazioni e le loro funzioni/scopi.

Glossario

- **Arma** = dotazione al personale di guardia.

  - *Sinonimi*: arma da fuoco, pistola.

- **A.T.C** = Il codice che permette ad ogni confezione di medicinale di essere riconosciuto e tracciato.

- **Condizione** = Malattia psicologica di cui soffre il paziente, diagnosticata dallo psichiatra.

  - *Sinonimi*: disturbo mentale, alienazione, demenza, follia, squilibrio.

- **Dispositivo medico** = Qualunque strumento atto ad utilizzo diagnostico/curativo/somministrativo per il trattamento o la cura di una condizione o malattia.

- **Malattia** = Malattia fisica dovuta a reazioni fisiche per infiammazioni, proliferazioni batteriche o virus.

  - *Sinonimi*: affezione, infezione, malessere, malore.

- **Frequenza** = Quantitativo di ore tra una somministrazione e l'altra.

- **Diagnosi**: elenco delle possibili malattie, legate a mancanza di salute fisica, che un check up può portare alla luce.

D.U.P =Data di uscita prevista.

D.D.N = Data di nascita.

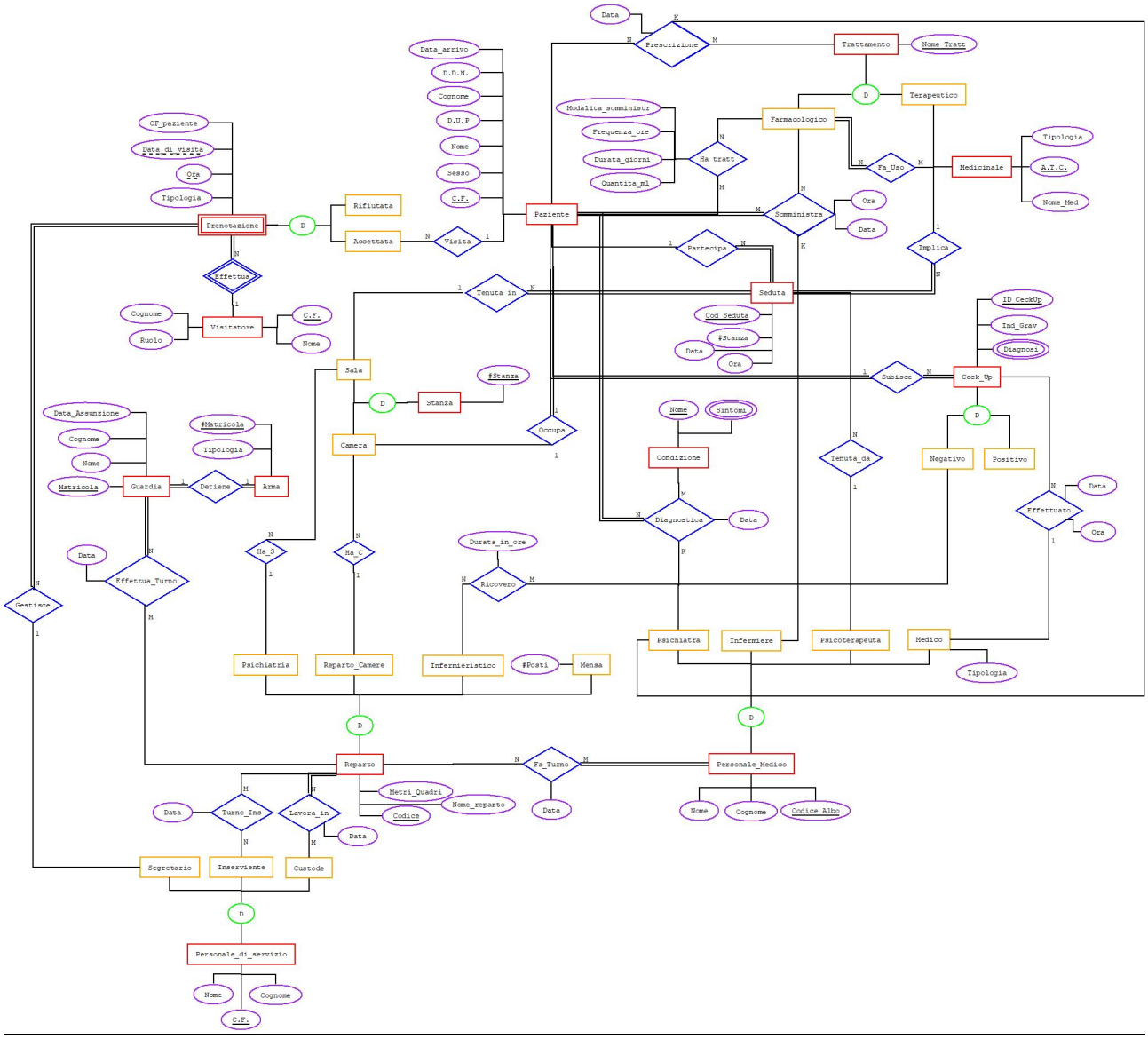
C.F. = Codice Fiscale.

Ind_grav = Indice di gravità.

Progettazione del database:

In questo capitolo verranno mostrate e descritte tutte le fasi di progettazione del database iniziando dal diagramma EE/R fino al Diagramma Relazionale sintetizzato dall'EE/R, analizzando per ognuno di essi le principali enti e caratteristiche con apposite sezioni di commento.

Diagramma EE/R:



Le entità sono evidenziate in rosso.

Le associazioni in blu.

Le specializzazioni in ocra.

Gli attributi in violetto.

Commento sulle entità:

Paziente: del paziente vogliamo registrare, il nome ed il cognome, la sua data di nascita e la sua possibile data di morte avvenuta nella struttura, il sesso ed il suo codice fiscale.

Stanza: è una entità che si specializza in camere e sale, rispettivamente associate al reparto camere ed la reparto psichiatrico. É identificata da un numero stanza, nella specializzazione di camera troviamo un attributo dotazione che identifica gli elementi come “Letto, bagno, lavandino ecc.”.

Condizione: ne memorizziamo il nome ed i suoi sintomi.

Trattamento: conserviamo il nome e la durata, esso può essere farmacologico e terapeutico, nel caso sia farmacologico fa uso de medicinali ed è somministrato dagli infermieri, se invece è terapeutico implica delle sedute fatte tra il paziente ed uno psicoterapeuta.

Seduta: di essa conserviamo il codice identificativo, il numero della stanza in cui viene effettuata, e la data e l’ora in cui si svolge.

Check Up: il check up viene effettuato periodicamente ad ogni paziente per controllarne lo stato di salute, si memorizza il suo id che lo identifica, l’indice di gravità e la diagnosi.

Medicinale: vogliamo conoscerne la tipologia il nome ed il suo C.I.U.

Reparto: del reparto vogliamo solo sapere il suo codice e la grandezza territoriale. Si specializza in 4 tipi, mensa, infermieristico, psichiatrico e camere, in ognuno di loro lavorano determinati dipendenti e se ne conservano caratteristiche peculiari come il numero delle camere per il reparto camere, il numero posti per la mensa, il numero sale per il reparto psichiatrico.

Personale Medico: riguarda a questa entità, ha come chiave il codice albo, che identifica univocamente ogni persona nel settore medico, nome e cognome, si specializza inoltre in psichiatra, infermiere, psicoterapeuta e medico, che svolgono specifiche mansioni.

Personale di Servizio: indentificate dal codice fiscale delle persone che ne fanno parte conserviamo nome e cognome. Anche questa entità si specifica tra segretari, inservienti e custodi. Come per l'entità precedente ognuna delle specializzazioni svolge una specifica mansione all'interno della struttura.

Visitatore: di un certo visitatore vogliamo sapere il motivo della visita e quindi il suo ruolo, nome, cognome ed il suo codice fiscale.

Prenotazione: la prenotazione viene effettuata da un visitatore e gestita da un segretario, di essa conserviamo la tipologia, la data, la visita ed il codice fiscale di chi deve essere visitato, inoltre, essa può essere accettata o rifiutata, per via dell'indisponibilità a ricevere visite perché occupato in sedute o Somministrazioni.

Guardia: delle guardie conserviamo i dati anagrafici, la data della sua assunzione ed utilizziamo la sua matricola militare per identificarla.

Arma: un'arma è posseduta da una sola guardia, memorizziamo il tipo e l'attributo identificante che è il numero matricola.

Commento sulle associazioni:

Iniziando dall'alto del diagramma EE/R troviamo:

Prescrizione: è un'associazione trivalente che serve a tenere traccia con corrispettiva data, di quando uno psicoterapeuta ha prescritto quale trattamento a quale paziente, la sua molteplicità è M ad N a K, poiché vi sono associazioni multiple da ogni lato ad ogni lato. Verrà implementata con una tabella di transizione dedicata.

Ha tratt: questa associazione serve a tenere traccia dei personali livelli, dosaggi e durate dei trattamenti farmacologici prescritti ai pazienti che ne hanno necessitato. Verrà implementata con una tabella di transizione dedicata. Molteplicità di tipo N ad M.

Somministra: tiene traccia delle somministrazioni giornaliere dei trattamenti farmacologici ai pazienti da parte degli infermieri. Implementata con una tabella di transizione. Molteplicità N M K.

Fa uso: tiene conto di quali farmaci fa uso un trattamento farmacologico, implementata con una tabella esterna. Molteplicità M ad N.

Implica: questa associazione aiuta a tenere traccia per ogni seduta, di quale trattamento terapeutico si è fatto uso. La molteplicità è di tipo 1 ad N e sarà implementata come chiave esterna nella tabella Seduta.

Partecipa: in combinazione con la precedente associa ad una seduta un solo paziente, per tenere traccia di che trattamento terapeutico sia stato somministrato. Molteplicità 1 ad N implementata nella tabella Seduta.

Tenuta in: ancora, in combinazione con le due precedenti serve a capire dove ogni singola seduta in che sala del reparto psichiatrico si sia tenuta. Implementata nella tabella Seduta poiché con molteplicità 1 ad N.

Visita: relazione che associa ad ogni prenotazione accettata la corrispettiva visita del visitatore al paziente interessato. Con molteplicità 1 ad N, viene implementata con una tabella dedicata.

Effettua: si ricollega alla visita discussa pocanzi, un visitatore può effettuare delle prenotazioni. Anch'essa implementata nella tabella Prenotazione poiché con una molteplicità 1 ad N.

Occupa: la relazione occupa associa ad ogni paziente una camera del reparto camere e ad ogni camera un paziente, viene implementata nella tabella Paziente, molteplicità 1 ad 1.

Subisce: questa relazione riguarda le visite mediche subite dal paziente. La molteplicità 1 ad N ne permette l'implementazione nella tabella Check-up.

Tenuta da: riguarda sempre la gestione e la conservazione dei dati riguardanti le seduta, serve a tener conto di quale psicoterapeuta ha svolto la seduta. Molteplicità 1 ad N implementata nella tabella Seduta.

Effettuato: associa il Check-up ad un Medico e viene registrata la data e l'ora della visita. La sua molteplicità è 1 ad N ma dovendo conservare anche la data e l'ora viene implementata in una tabella transitoria dedicata.

Diagnostica: altra associazione trivalente che ci aiuta a tener traccia di quale psichiatra abbia diagnosticato quale condizione a quale paziente. La molteplicità è N M K e verrà pertanto implementata come tabella transitoria dedicata.

Ricovero: quella che alla fine sarà una tabella di transizione con un attributo che registra la durata di ore di ricovero, serve a registrare, se il paziente ha ricevuto Check-Up Negativo, per quanto tempo questi dovrà rimanere sotto osservazione nel reparto Infermieristico.

Ha c: associazione che registra quali stanze sono possedute dal reparto Camere. 1 ad N implementata nella tabella Stanza.

Ha s: associazione che registra quali stanze sono possedute dal reparto psichiatrico. 1 ad N implementata nella tabella Stanza.

Detiene: è una associazione che collega 1 arma ad 1 guardia. Verrà implementata nella tabella Arma.

Effettua turno: di tipo M ad N e con un attributo che conserva la data, associa le guardie al reparto in cui svolgono il turno giornaliero all'interno della struttura.

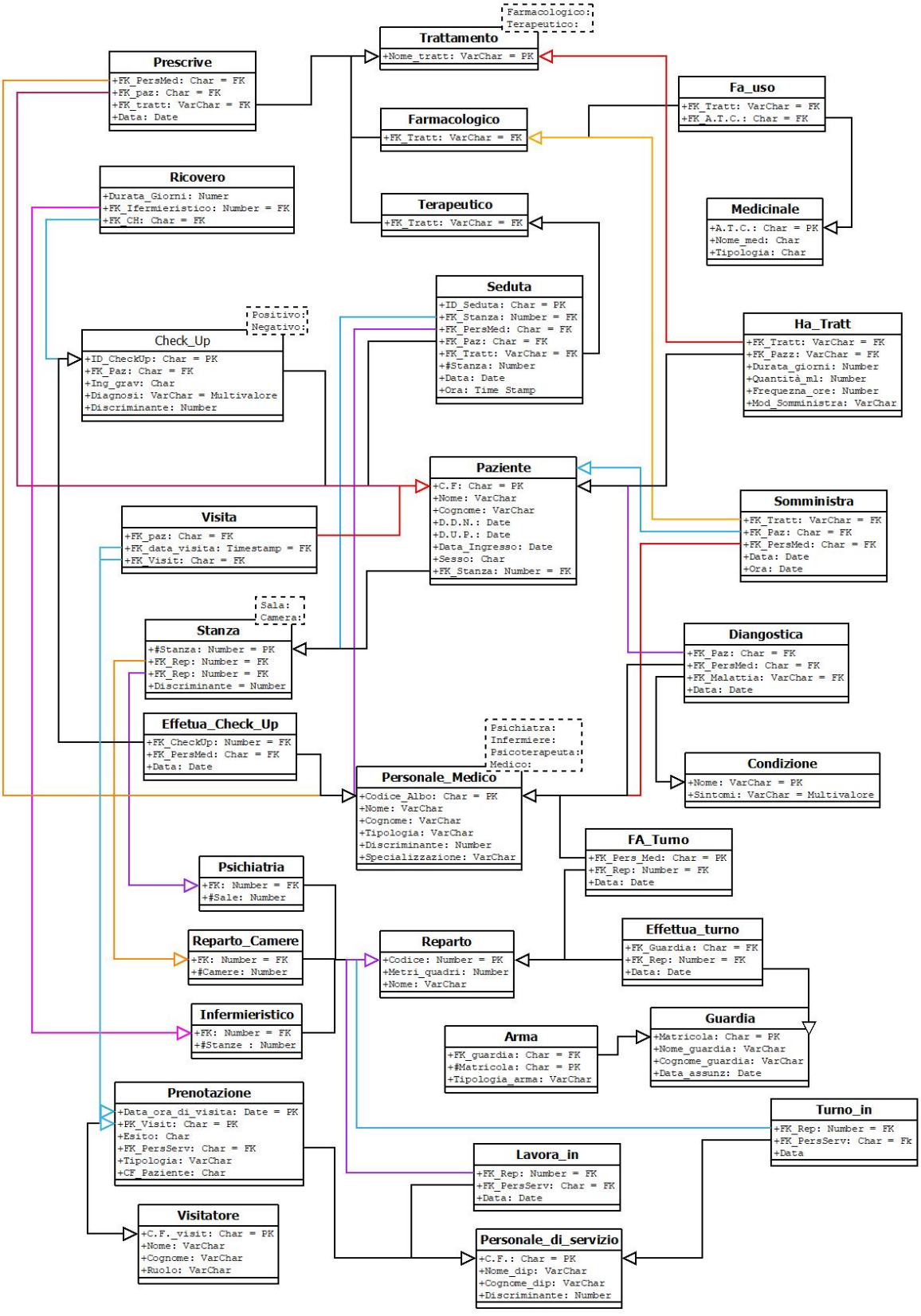
Gestisce: tra la specializzazione Segretario del personale di servizio e le Prenotazioni effettuate dai visitatori, vige questa associazione che registra quali prenotazioni sono state gestite da quale segretario. 1 ad M implementata nella tabella Prenotazione.

Turno Ins: associazione con molteplicità M ad N che registra la data in cui un Inserviente svolge il suo turno nel reparto X.

Lavora in: associazione con molteplicità M ad N che registra la data in cui un Custode svolge il suo turno nel reparto X.

Fa turno: associazione con molteplicità M ad N che registra la data in cui una persona del personale medico svolge il suo turno nel reparto X.

Diagramma Relazionale:



Commento sulle Tabelle:

Le tabelle presentate nella pagina precedente sono la diretta conseguenza e traduzione delle entità ed associazioni del diagramma EE/R. Poiché vi è già stato descritto nelle pagine precedenti la natura delle entità e delle relazioni i gioco in questa pagina verranno solo riassunte le principali enti del diagramma.

Inizieremo con l'elencare le tabelle delle entità esplicitando in quali tra esse vi siano vincoli di dipendenza, casi in cui si stato deciso di implementarvi le associazioni. Troviamo dunque:

- *Trattamento*: Classe madre di Farmacologico e Terapeutico, implementate come tabelle separate seguendo il principio del posizionamento verticale, quindi con chiavi esterne che si riferiscono alla classe madre.
- *Medicinale*: traduzione diretta dell'entità Medicinale nel diagramma EE/R.
- *Seduta*: questa entità comprende diverse chiavi esterne che si riferiscono alle 4 tabelle da cui dipende, Paziente, Terapeutico, Personale Medico e Stanza, le associazioni 1 ad N di cui si è discusso nel commento sulle associazioni, più i principali attributi come la chiave "ID_Seduta"
- *Check-Up*: vive di una chiave esterna verso il paziente, per l'implementazione dell'associazione Subisce.
- *Paziente*: tabella che possiede gli attributi di "Paziente" più una chiave esterna che la ricollega a Stanza, per via dell'implementazione Occupa precedentemente discussa.
- *Stanza*: la quale è una classe madre implementata come tabella unica con discriminante data la totale assenza di attributi nelle classi figlie Camera e Sala. Possiede due chiavi esterne che la ricollegano al reparto Camere ed a reparto Psichiatrico per implementare le associazioni "Ha c" ed "Ha s" discusse nel commento sulle associazioni.
- *Condizione*: implementata algoritmicamente dal diagramma EE/R.
- *Reparto*: implementazione diretta anche in questo caso.
- *Guardia*: va come le precedenti tre tabelle.
- *Visitatore*: implementazione diretta.
- *Personale di servizio*: classe madre per segretario, custode ed inserviente, l'implementazione è sotto tabella unica, l'aggiunta del discriminante per identificare il ruolo è pertanto giustificata.
- *Personale Medico*: classe madre per il personale medico, è stata implementata come tabella unica nella quale collassano le classi figlie, vi è un discriminante per distinguere le sottoclassi.
- *Prenotazione*: oltre ai suoi banali attributi, vi troviamo una chiave esterna verso il visitatore, per implementare l'associazione Effettua 1 ad N.
- *Arma*: ultima entità che troviamo rappresentata nel diagramma relazionale possiede oltre che i suoi attributi, anche una chiave esterna che la ricollega alla guardia che la possiede, è di fatto l'implementazione dell'associazione 1 ad 1 Detiene.

Riguardo invece alle tabelle non discusse, possiamo dire che **fanno parte di un unico gruppo**, quello delle **tabelle transitorie**, implementazioni delle associazioni multiple su ogni cardinalità, che possiedono personali attributi di cui tener traccia. Esse sono:

- *Prescrive*: chiave esterna verso Trattamento, Paziente e Personale Medico, data.
- *Ricovero*: chiave esterna verso il Reparto infermieristico e verso il Check-Up, durata ricovero.
- *Fa uso*: chiave esterna verso Farmacologico e Medicinale.
- *Ha tratt*: chiave esterna verso il Farmacologico e Paziente, durata trattamento in giorni, quantità in ml somministrate, frequenza in ore, modalità di somministrazione.

- *Somministra*: chiave esterna verso Farmacologico, Paziente e Personale medico, data ed ora.
- *Diagnostica*: chiave esterna verso Paziente, Personale medico e Condizione, data.
- *Effettua Check-Up*: chiave esterna verso Check-Up, Personale medico data ed ora.
- *Fa Turno*: chiave esterna verso Personale medico, Reparto e data turno.
- *Effettua Turno*: chiave esterna verso Guardia, verso Reparto e data turno.
- *Turno in*: chiave esterna verso Personale di Servizio, Reparto e data turno.
- *Lavora In*: chiave esterna verso Personale di servizio, Reparto e data turno.
- *Visita*: chiave esterna composta verso la chiave esterna composta di Prenotazione, chiave esterna verso Paziente

Valutazione delle forme normali:

Prima forma normale (1NF):

La valutazione dello schema riguardo la prima forma normale è rispettata completamente fatta eccezione per due casi riguardante Condizione, con il suo attributo sintomi, e l'entità Check-Up con l'attributo diagnosi. Poiché entrambi multivalore per definizione non rispettano la prima forma normale, la quale ci impone che tutti gli attributi presenti debbano essere atomici. Sono stati lasciati in tale modalità poiché ritenuti di fondamentale importanza per la definizione delle entità a cui appartengono; una condizione psicotica possiede diversi sintomi associati e ad un Check-Up medico possono risultare diverse malattie diagnosticabili/infiammazioni ecc. le quali devono essere registrate in qualche modo.

Seconda forma normale (2NF):

La seconda forma normale richiede che sia rispettata la 1NF ed indica come regola quella di non avere attributi indipendenti dalla Chiave dell'entità o dipendenti da essa solo parzialmente. La seconda forma normale è rispettata nello schema completamente tranne per le entità precedentemente descritte che non rispettano la prima forma normale.

Terza forma normale e BCNF (3NF, BCNF):

Così come nel caso della 2NF, anche la Terza per esistere deve avere il presupposto che sia rispettata la 2NF e. Troviamo qui due problemi nelle entità Medicinale e Reparto. Riguardo alla prima dal suo attributo non chiave *Nome_med (Nome Medicinale)* possiamo risalire ad un altro suo attributo non chiave *Tipologia,* esempio, "Tachipirina -> Antidolorifico" oppure "Ibuprofene -> Antinfiammatorio".

Riguardo invece all'entità Reparto, dal suo attributo *Nome_rep (Nome Reparto)* possiamo risalire ad un altro suo attributo *Metri_quadri*, dato che ad un nome reparto solitamente univoco, viene associata una sua estensione.

Possiamo pertanto concludere che la 3NF non sia rispettata prendendo anche in considerazione le entità Condizione e Check-Up.

La BCNF afferma che una tabella è in BCNF se, per ogni dipendenza funzionale non banale X -> Y, dove X rappresenta un insieme di attributi e Y rappresenta un altro insieme di attributi, la chiave primaria determina completamente ogni attributo non chiave nella tabella. In altre parole, se ogni dipendenza funzionale non banale in una tabella è determinata completamente dalla chiave primaria, allora la tabella è in BCNF.

Nel nostro caso quindi lo schema non si trova in BCNF.

Implementazione SQL:

In questo capitolo invece verranno analizzati i passaggi ed il codice implementativo per portare in vita gli schemi e le relazioni analizzate nel capitolo precedente.

Si parlerà di utenti e permessi, codice implementativo per le tabelle, implementazione dei vincoli dinamici per l'integrità dei dati, procedure annesse agli utenti ed infine verranno fatti dei campionamenti dal popolamento ed analizzati gli inserimenti effettuati.

Utenti e permessi:

Al database qui proposto avranno accesso diversi utenti, ognuno dei quali ha dei permessi di esecuzione ed accesso al database diversi dagli altri e personali, fatta eccezione dell'amministratore il quale possiede tutti i permessi.

Gli utenti presenti nel database saranno di conseguenza le figure con maggior importanza, di rilievo e che devono gestire operazioni importanti, comunque inerenti al loro campo lavorativo. Troviamo:

- Segretari: gestiscono diverse tabelle nel database, come quelle dei turni di ognuna delle persone che lavorano nella struttura, in particolare possono fare resoconti di giornata riguardo a cosa hanno fatto i pazienti ed hanno tutti i permessi necessari per gestire le prenotazioni dei visitatori
- Psichiatra: facente parte del personale medico, questa figura ha bisogno di avere accesso esclusivo ad informazioni sensibili riguardanti i pazienti, a lui è permesso l'inserimento nel database delle prescrizioni dei trattamenti, così come quello per l'inserimento delle Diagnostiche dele condizioni psichiche.
- Psicoterapeuta: molto vicino alla figura precedente troviamo lo psicoterapeuta, possiede solo due permessi, l'esecuzione della procedura per osservare i medicinali prescritti ad un Paziente e la gestione delle Sedute.
- L'Infermiere: anche lui con solo due permessi molto coerenti tra loro, può osservare i medicinali prescritti ai pazienti che poi somministra ai singoli, ha di fatto accesso alla tabella Somministrazione.
- Medico: il medico ha i permessi su tutta quella branca di operazioni e strutture che riguardano i Check-Up, quindi la loro registrazione, la prescrizione di ricoveri in infermeria la vista dei medicinali prescritti.
- Admin: il Data-Base Administrator è colui che gestisce tutto e tutto il resto, situazioni o tabelle più statiche e che non conservano informazioni critiche o delicate riguardante i Pazienti.

Implementazione degli utenti in SQL:

--Amministratore

CREATE USER AMMINISTRATORE_GENERALE IDENTIFIED BY ADMIN;

GRANT CONNECT, RESOURCE, DBA, TO AMMINISTRATORE_GENERALE;

GRANT ALL PRIVILEGES TO AMMINISTRATORE_GENERALE;

--Segretario

CREATE USER SEGRETARIO IDENTIFIED BY PASS_SEGRETARIO;

GRANT EXECUTE ON Prenotazione_accettata TO SEGRETARIO;

GRANT EXECUTE ON Resoconto_giornata TO SEGRETARIO;

GRANT EXECUTE ON Lavoratori_nel_reparto TO SEGRETARIO;

GRANT EXECUTE ON Turni_personale_medico TO SEGRETARIO;

GRANT INSERT ON Turno_in TO SEGRETARIO;

GRANT INSERT ON Lavora_in TO SEGRETARIO;

GRANT INSERT ON Effettua_turno TO SEGRETARIO;

GRANT INSERT ON Prenotazione TO SEGRETARIO;

GRANT INSERT ON Visitatore TO SEGRETARIO;

--Psichiatra

CREATE USER PSICHIATRA IDENTIFIED BY PASS_PSICHIATRA;

GRANT EXECUTE ON Medicinali_del_paziente TO PSICHIATRA;

GRANT INSERT INTO Diagnostica TO PSICHIATRA;

GRANT INSERT INTO Prescrizione TO PSICHIATRA;

--Psicoterapeuta

CREATE USER PSICOTERAPEUTA IDENTIFIED BY PASS_PSICOTERAPEUTA;

GRANT EXECUTE ON Medicinali_del_paziente TO PSICOTERAPEUTA;

GRANT INSERT ON Seduta TO PSICOTERAPEUTA;

--Infermiere

CREATE USER INFERMIERE IDENTIFIED BY PASS_INFERMIERE;

GRANT EXECUTE ON Medicinali_del_paziente TO INFERMIERE;

```
GRANT INSERT ON Somministra TO INFERMIERE;


--Medico
CREATE USER MEDICO IDENTIFIED BY PASS_MEDICO;
GRANT EXECUTE ON Medicinali_del_paziente TO MEDICO;
GRANT EXECUTE ON Ricovero_in_infermieria TO MEDICO;
GRANT INSERT ON Effettua_ck TO MEDICO;
GRANT INSERT ON Check_Up TO MEDICO;
```

Implementazione SQL delle tabelle:

Nel capitolo precedente sono state definite le tabelle del diagramma relazionale, al passaggio in linguaggio Oracle SQL non vi è nulla di criptico o diverso da come il database viene rappresentato nel diagramma, quello che di diverso accade è che vi troviamo la presenza di diversi vincoli per gli attributi, non vengono a mancare durante la creazione delle singole tabelle, "NOT NULL", per obbligare nell'inserimento di alcuni dati sensibili, i "CHECK IN" per definire domini di esistenza di particolari attributi e "UNIQUE" per identificare ogni valore di un attributo di una tabella univoco e non ripetibile.

Tabella trattamento: non vi troviamo nulla tranne il nome che ne fa da chiave:

```
CREATE TABLE Trattamento(
   Nome_tratt      VARCHAR(40) PRIMARY KEY
);
```

Tabella figlia per il trattamento farmacologico:

```
CREATE TABLE Farmacologico(
   Nome_tratt_f     VARCHAR(40) PRIMARY KEY,

   --Chiave esterna verso calsse madre Trattamento
   CONSTRAINT FK_farmacologico_figlia
   FOREIGN KEY (Nome_tratt_f) REFERENCES Trattamento(Nome_tratt)  ON DELETE CASCADE
);
```

Tabella figlia per il trattamento terapeutico:

```
CREATE TABLE Terapeutico(
   Nome_tratt_t  VARCHAR(40) PRIMARY KEY,

   --Chiave esterna verso classe madre Trattamento
   CONSTRAINT FK_terapeutico_figlia
   FOREIGN KEY (Nome_tratt_t) REFERENCES Trattamento(Nome_tratt) ON DELETE CASCADE
);
```

Tabella Medicinale

```
CREATE TABLE Medicinale(
   ATC        CHAR(15) PRIMARY KEY,
   Nome_med    VARCHAR(30) NOT NULL,
   Tipologia   VARCHAR(30) NOT NULL
);
```

Tabella Fa_uso, transitoria tra Medicinale e trattamento farmacologico:

```
CREATE TABLE Fa_uso(
   Nome_tratt_uso    VARCHAR(40) NOT NULL,
   ATC_usato         CHAR(15)NOT NULL,

   --Chiave esterna verso Trattamento
   CONSTRAINT FK_fa_uso_trattamento
   FOREIGN KEY (Nome_tratt_uso) REFERENCES Trattamento(Nome_tratt),

   --Chiave esterna verso Medicinale
   CONSTRAINT FK_fa_uso_medicinale
   FOREIGN KEY (ATC_usato) REFERENCES Medicinale(ATC)
);
```

Tabella Reparto:

```
-- Tabella reparto
CREATE TABLE Reparto (
   codice_rep        NUMBER PRIMARY KEY,
   Nome_reparto      VARCHAR(30) NOT NULL,
   Metri_quadri      NUMBER
);
```

Tabella reparto Mensa figlia di reparto:

```
CREATE TABLE Mensa (
    codice_m          NUMBER,
    Numero_posti      NUMBER,

    --Chiave esterna verso Reparto
    CONSTRAINT FK_mensa_reaprto
    FOREIGN KEY (codice_m) REFERENCES Reparto(codice_rep)
);
```

Tabella reparto Infermieristico figlia di reparto:

```
-- Tabella sottotipo infermieristico
CREATE TABLE Infermieristico (
    codice_rep_i     NUMBER PRIMARY KEY,
    Numero_stanze    NUMBER NOT NULL,

    --Chiave esterna verso classe madre Reparto
    CONSTRAINT FK_infermieristico_figlia
    FOREIGN KEY (codice_rep_i) REFERENCES reparto(codice_rep) ON DELETE CASCADE
);
```

Tabella reparto Psichiatrico figlia di reparto:

```
-- Tabella sottotipo psichiatrico
CREATE TABLE Psichiatrico (
    codice_rep_p    NUMBER PRIMARY KEY,
    numero_sale     NUMBER NOT NULL,

    --Chiave esterna verso classe madre Reparto
    CONSTRAINT FK_psichiatrico_figlia
    FOREIGN KEY (codice_rep_p) REFERENCES reparto(codice_rep) ON DELETE CASCADE
);
```

Tabella reparto Camere figlia di reaprto:

```
-- Tabella sottotipo camere
CREATE TABLE Camere (
    codice_rep_c    NUMBER PRIMARY KEY,
    numero_camere   NUMBER NOT NULL,

    --Chiave esterna verso classe madre Reparto
    CONSTRAINT FK_camere_figlia
    FOREIGN KEY (codice_rep_c) REFERENCES reparto(codice_rep) ON DELETE CASCADE
);
```

Tabella madre Stanza nella quale collassano le classi figlie 'Camera' e 'Sala' sotto forma di discriminante:

```
CREATE TABLE Stanza(
   Numero_stanza       NUMBER PRIMARY KEY,
   codice_rep_cam      NUMBER,
   codice_rep_s        NUMBER,
   Discriminante_s     NUMBER NOT NULL CHECK (Discriminante_s IN (1,2)),

   --Chiave esterna verso Reparto camere
   CONSTRAINT FK_stanza_camere
   FOREIGN KEY (codice_rep_cam) REFERENCES Camere(codice_rep_c),

   --Chiave esterna verso Reparto psichiatrico
   CONSTRAINT FK_stanza_psichiatrico
   FOREIGN KEY (codice_rep_s) REFERENCES Psichiatrico(codice_rep_p)
);
```

Tabella del Personale Medico, in questa tabella collassano le classi figlie Infermiere, Psichiatra, Psicoterapeuta, Medico. L'attributo 'Tipologia' di 'Medico' si presenta come NULL se il discriminante è diverso da 4 (implementato tramite Trigger):

```
CREATE TABLE Personale_Medico(
   Codice_albo          CHAR(15) PRIMARY KEY,
   Nome                 VARCHAR(20) NOT NULL,
   Cognome              VARCHAR(20) NOT NULL,
   Tipologia            VARCHAR(30) NOT NULL,
   Specializzazione     VARCHAR(30),
   Discriminante        NUMBER NOT NULL CHECK (Discriminante IN (1,2,3,4))

);
```

Tabella Fa_turno transitoria tra il Personale Medico e Reparto:

```
CREATE TABLE Fa_Turno(
   pers_med          CHAR(15),
   codice_rep_turno    NUMBER,
   data_turno         DATE NOT NULL,

   --Chiave esterna verso Personale_medico
   CONSTRAINT FK_turno_persmed
   FOREIGN KEY (pers_med) REFERENCES Personale_Medico(Codice_albo),

   --Chiave esterna verso Reparto
   CONSTRAINT FK_turno_reparto
   FOREIGN KEY (codice_rep_turno) REFERENCES Reparto(codice_rep)
);
```

Tabella dei Pazienti:

```
CREATE TABLE Paziente(
   CF               CHAR(16) PRIMARY KEY,
   Numero_stanza_paz   NUMBER NOT NULL,
   Nome             VARCHAR(20) NOT NULL,
   Cognome           VARCHAR(20) NOT NULL,
   DDN              DATE NOT NULL,
   DUP              DATE,
   Data_ingresso       DATE NOT NULL,
   Sesso            CHAR(1) CHECK(Sesso IN ('M','F')),

   --Chiave esterna verso Stanza
   CONSTRAINT FK_paziente_stanza
   FOREIGN KEY (Numero_stanza_paz) REFERENCES Stanza(Numero_stanza));
```

Tabella Ha_tratt, transitoria tra Paziente e trattamento farmacologico:

```
CREATE TABLE Ha_tratt(
    Nome_tratt_fatto    VARCHAR(40) CHECK( Nome_tratt_fatto IN ('Terapia antidepressiva','Terapia ansiolitica','Terapia Stabilizzatore_dell_umore','Terapia antipsicotica')),
    CF_paziente_farm    CHAR(16),
    Durata_giorni       NUMBER NOT NULL,
    Quantita_ml         NUMBER NOT NULL,
    Frequenza_ore       NUMBER NOT NULL,
    Modalita_sommin     VARCHAR(40) NOT NULL,

    --Chiave esterna verso Tratt_Farmacologico
    CONSTRAINT FK_paz_tratt
    FOREIGN KEY (CF_paziente_farm) REFERENCES Paziente(CF),

    --Chiave esterna verso
    CONSTRAINT FK_farm_tratt
    FOREIGN KEY (Nome_tratt_fatto) REFERENCES Farmacologico(Nome_tratt_f)
);
```

Tabella Prescrizione, transitoria e trivalente tra il Paziente il Trattamento e lo Psichiatra:

```
CREATE TABLE Prescrizione (
    CF_paz_prescrizione         CHAR(16),
    Codice_albo_prescrizione    CHAR(15),
    Nome_tratt_prescritto       VARCHAR(40),
    data_prescrizione           DATE,

    --Chiave esterna verso Pazietne
    CONSTRAINT FK_prescrizione_trattamento
    FOREIGN KEY (CF_paz_prescrizione) REFERENCES Paziente(CF),

    --Chiave esterna verso Personale Medico
    CONSTRAINT FK_prescrizione_persmed
    FOREIGN KEY (Codice_albo_prescrizione) REFERENCES Personale_Medico(Codice_albo),

    --Chiave esterna verso Trattamento
    CONSTRAINT FK_prescrizione_trattamenmto
    FOREIGN KEY (Nome_tratt_prescritto) REFERENCES Trattamento(Nome_tratt)
);
```

Tabella per Seduta:

```
CREATE TABLE Seduta(
   ID_seduta         char(10) PRIMARY KEY,
   Numero_stanza_s   NUMBER,
   Codice_albo_s     CHAR(15),
   CF_s              CHAR(16),
   Nome_tratt_s      VARCHAR(40),
   Data_ora_s        TIMESTAMP NOT NULL,

   --Chiave esterna verso Stanza
   CONSTRAINT FK_seduta_stanza
   FOREIGN KEY (Numero_stanza_s) REFERENCES Stanza(Numero_stanza),

   --Chiave esterna verso Personale medico
   CONSTRAINT FK_seduta_persmed
   FOREIGN KEY (Codice_albo_s) REFERENCES Personale_Medico(Codice_albo),

   --Chiave esterna verso Paziente
   CONSTRAINT FK_seduta_paziente
   FOREIGN KEY (CF_s) REFERENCES Paziente(CF),

   --Chiave esterna verso Trattamento
   CONSTRAINT FK_seduta_trattamento
   FOREIGN KEY (Nome_tratt_s) REFERENCES Terapeutico(Nome_tratt_t)
);
```

Tabella Somministra transitoria e trivalente tra Paziente, Infermiere e Trattamento Farmacologico:

```
CREATE TABLE Somministra(
   Nome_tratt_somm    VARCHAR(40),
   CF_somm            CHAR(16),
   Codice_albo_somm   CHAR(15),
   Data_ora_somm      TIMESTAMP NOT NULL,

   --Chiave esterna verso Trattamento
   CONSTRAINT FK_somministra_farmacologico
   FOREIGN KEY (Nome_tratt_somm) REFERENCES Farmacologico(Nome_tratt_f),

   --Chiave esterna verso Paziente
   CONSTRAINT KF_somministra_paziente
   FOREIGN KEY (CF_somm) REFERENCES Paziente(CF),

   --Chiave esterna verso Personale medico
   CONSTRAINT FK_somministra_persmed
   FOREIGN KEY (Codice_albo_somm) REFERENCES Personale_Medico(Codice_albo)
);
```

Tabella Check-Up:

```
CREATE TABLE Check_Up(
   ID_CheckUp     CHAR(15) PRIMARY KEY,
   CF_checkup     CHAR(16),
   risultato      CHAR(1) CHECK (risultato IN ('P', 'N')),
   Ind_grav       NUMBER CHECK (Ind_grav IN(0,1,2,3,4,5)),

   --Chiave esterna verso Paziente
   CONSTRAINT FK_check_paz
   FOREIGN KEY (CF_checkup) REFERENCES Paziente(CF) );
```

Tabella Diagnosi per esprimere l'attributo multivalore 'Diagnosi' della tabella Check-Up:

```
CREATE TABLE Diagnosi(
   ID_CheckUp              Char(15),
   Malattia_diagnosticata   VARCHAR(30),

   --Chiave esterna della tabella per i multivalori verso Check_Up
   CONSTRAINT FK_diagnosi_ck
   FOREIGN KEY (ID_CheckUp) REFERENCES Check_Up(ID_CheckUp)
);
```

Tabella Effettua_CK (Effettua Check-Up) transitoria tra Check-Up e Medico:

```
CREATE TABLE Effettua_ck(
   data_ora_ck     TIMESTAMP NOT NULL,
   ck_up_eff       CHAR(15),
   Codice_albo_ck  CHAR(15),

   --Chiave esterna verso Personale Medico
   CONSTRAINT FK_ck_persmed
   FOREIGN KEY (Codice_albo_ck) REFERENCES Personale_Medico(Codice_albo),

   --Chiave esterna verso Check_up
   CONSTRAINT FK_check_up
   FOREIGN KEY (ck_up_eff) REFERENCES Check_Up (ID_CheckUp)
);
```

Tabella Ricovero transitoria tra Check-Up e Reparto Infermiristico:

```
CREATE TABLE Ricovero (
   giorni_durata       NUMBER NOT NULL,
   ricovero_inferm     NUMBER,
   ricovero_checkup    CHAR(15),

   --Chiave esterna verso Reparto infeermieristico
   CONSTRAINT FK_ricovero_inf
   FOREIGN KEY (ricovero_inferm) REFERENCES Infermieristico(codice_rep_i),

   --Chiave esterna verso Check_up
   CONSTRAINT FK_ricovero_chekup
   FOREIGN KEY (ricovero_checkup) REFERENCES Check_Up(ID_CheckUp)
);
```

Tabella Condizione:

```
CREATE TABLE Condizione(
   Nome    VARCHAR(60) PRIMARY KEY
);
```

Tabella per implementare l'attributo multivalore 'Sintomi' della tabella Condizione:

```
CREATE TABLE Sintomi(
   sintomo_diagn        VARCHAR(60),
   fk_condizione        VARCHAR(60),

   --Chiave esterna dalla tabella dei Sintomi verso Condizione
   CONSTRAINT FK_Sintomi_condizione
   FOREIGN KEY (fk_condizione) REFERENCES Condizione(Nome)
);
```

Tabella Diagnostica trivalente e transitoria tra Paziente, Psichiatra e Condizione:

```
CREATE TABLE Diagnostica(
   CF_dia            CHAR(16),
   Codice_albo_dia   CHAR(15),
   Nome_condizione   VARCHAR(60),
   data_diagn        TIMESTAMP,

   --Chiave esterna verso Paziente
   CONSTRAINT FK_diagnostica_paziente
   FOREIGN KEY (CF_dia) REFERENCES Paziente(CF),

   --Chiave esterna verso Personale Medico
   CONSTRAINT FK_diagnostica_persmed
   FOREIGN KEY (Codice_albo_dia) REFERENCES Personale_Medico(Codice_albo),

   --Chiave esterna verso Malattia
   CONSTRAINT FK_diagnostica_condizione
   FOREIGN KEY (Nome_condizione) REFERENCES Condizione(Nome)
);
```

Tabella per il personale di Guardia:

```
CREATE TABLE Guardia(
   Matricola      CHAR(20) PRIMARY KEY,
   nome_g         VARCHAR(20) NOT NULL,
   cognome_g      VARCHAR(20) NOT NULL,
   data_assunzione DATE NOT NULL
);
```

Tabella Arma, possedute dalle guardie:

```
CREATE TABLE Arma(
   Numero_matricola    NUMBER PRIMARY KEY,
   matricola_guardia   CHAR(20) UNIQUE,
   Tipologia           VARCHAR(20) NOT NULL,

   --Chiave esterna verso Guardia
   CONSTRAINT FK_arma_guardia
   FOREIGN KEY (matricola_guardia) REFERENCES Guardia(Matricola)
);
```

Tabella Effettua_Turno per associare le Guardie ad un reparto in una certa data:

```
CREATE TABLE Effettua_turno(
   Matricola_turno     CHAR(20),
   Cod_rep_turno_g     NUMBER,
   data_turno_g        DATE NOT NULL,

   --Chiave esterna verso Guardia
   CONSTRAINT FK_turno_guardia
   FOREIGN KEY (Matricola_turno) REFERENCES Guardia(Matricola),

   --Chiave esterna verso Reparto
   CONSTRAINT FK_turno_rep_g
   FOREIGN KEY (Cod_rep_turno_g) REFERENCES Reparto(codice_rep)
);
```

Tabella Personale_di_Servizio, in essa collassano le classi figlie Inserviente, Segretario, Custode sotto forma di discriminante:

```
CREATE TABLE Personale_di_servizio(
   CF_Pers_serv            CHAR(16) PRIMARY KEY,
   Nome_dip                VARCHAR(20) NOT NULL,
   Cognome_dip             VARCHAR(20) NOT NULL,
   discriminante_pers_serv  NUMBER NOT NULL CHECK ( discriminante_pers_serv IN(1,2,3))
);
```

Tabella Turno_in transitiva tra Personale di Servizio (Inserviente) e Reparto:

```
CREATE TABLE Turno_in(
   CF_inserviente_turno CHAR(16),
   codice_rep_inser     NUMBER,
   data_turno           DATE NOT NULL,

   --Chiave esterna verso Inserviente
   CONSTRAINT FK_turno_inserviente
   FOREIGN KEY (CF_inserviente_turno) REFERENCES Personale_di_servizio(CF_Pers_serv),

   --Chiave esterna verso Reparto
   CONSTRAINT FK_turno_ins_rep
   FOREIGN KEY (codice_rep_inser) REFERENCES reparto(codice_rep)
);
```

Tabella Lavora_in transitiva tra Personale di Servizio (Custode) e Reparto:

```
CREATE TABLE Lavora_in(
   CF_custode_turno    CHAR(16),
   codice_rep_custode  NUMBER,
   data_turno          DATE NOT NULL,

   --Chiave esterna verso Custode
   CONSTRAINT FK_lavora_custode
   FOREIGN KEY (CF_custode_turno) REFERENCES Personale_di_servizio(CF_Pers_serv),

   --Chiave esterna verso Reparto
   CONSTRAINT FK_lavora_reaparto
   FOREIGN KEY (codice_rep_custode) REFERENCES reparto(codice_rep)
);
```

Tabella Visitatore:

```
CREATE TABLE Visitatore(
   CF_visitatore       CHAR(16) PRIMARY KEY,
   nome_visitatore     VARCHAR(20) NOT NULL,
   cog_visitatore      VARCHAR(20) NOT NULL,
   ruolo               VARCHAR(20) NOT NULL CHECK (ruolo IN ('Parente','Ricercatore','Medico'))
);
```

Tabella Prenotazione, debole rispetto a visitarore:

```
CREATE TABLE Prenotazione(
   data_ora_visita     TIMESTAMP,
   CF_dipendenza       CHAR(16),
   CF_pers_serv_pre    CHAR(16),
   CF_paziente         CHAR(16),
   esito               CHAR(16) CHECK (esito IN('Accettata','Rifiutata')),
   tipologia           VARCHAR(20) NOT NULL CHECK (tipologia IN ('Parentela','Ricerca','Medica')),

   --Definizione della chiave mista
   CONSTRAINT PK_Prenotazione PRIMARY KEY (CF_dipendenza, data_ora_visita),

   --Chiave esterna verso l'entitá forte
   CONSTRAINT FK_prenotazione_visitatore
   FOREIGN KEY (CF_dipendenza) REFERENCES Visitatore(CF_visitatore),

   --Chiave esterna verso Personale di servizio
   CONSTRAINT FK_prenotazione_personale
   FOREIGN KEY (CF_pers_serv_pre) REFERENCES Personale_di_servizio(CF_Pers_serv)
);
```

Tabella Visita transitiva tra le prenotazioni solo accettate (implementazione tramite trigger) ed il Paziente interessato:

```
CREATE TABLE Visita (
    data_visita_prenotazione    TIMESTAMP,
    CF_prenotazione_visita      CHAR(16),
    CF_visita_paziente      CHAR(16),

    --Chiave esterna verso Prenotazione
    CONSTRAINT visita_prenotazione
    FOREIGN KEY (CF_prenotazione_visita, data_visita_prenotazione) REFERENCES
Prenotazione(CF_dipendenza, data_ora_visita),

    --Chiave esterna verso Paziente
    CONSTRAINT visita_paziente
    FOREIGN KEY (CF_visita_paziente) REFERENCES Paziente(CF)
);
```

Implementazione dei Trigger per vincoli statici:

Nonostante il database fin ora progettato sia praticamente perfetto ed immacolato, vi troviamo comunque delle falle di sistema dovute a regole del mondo reale che non possono essere espresse nel diagramma EE/R, Diagramma relazionale o con la semplice aggiunta di vincoli dinamici alle tabelle, regole che non possono essere infrante o che di norma nel mondo reale vengono rispettate ma che all'interno del Database così come progettato fino ad ora possono essere infrante in qualunque modo, In ordine troviamo il trigger per impedire che:

1. A chiunque non sia Psichiatra di prescrivere trattamenti.
2. Persone diverse da Infermieri effettuino Somministrazioni.
3. Di dare ricoveri a pazienti che stanno bene.
4. I Check-Up non vengano effettuati da persone che non siano Medici.
5. Persone diverse da psichiatri tengano le sedute.
6. Ae sedute Psicoterapeutiche si tenga in una camera di un Paziente.
7. Ad un Paziente venga associata una Sala per le sedute invece che una Camera.
8. Le diagnosi vengano effettuate da qualcuno che non sia uno Psichiatra.
9. Custodi o Segretari effettuino turni di pulizia.
10. Segretari oppure Inservienti custodiscano i reparti.
11. Ad ogni individuo del Personale Medico venga associato un turno in un reparto che non è di sua competenza.
12. Venga inserita una Specializzazione nella tabella Personale Medico se, la persona in questione non è un medico.
13. Un Check-Up risultato Positivo, quindi dove il paziente sta bene, abbia un indice di gravità diverso da 0.
14. Un visitatore effettui una visita ad un Paziente se questa gli è stata rifiutata.
15. Persone minorenni entrino in struttura
16. Venghino create stanze ambigue, che appartengono cioè sia al reparto Camere che al reparto Psichiatrico (Se è una camera viene associata al reparto Camere, se è una sala al reparto Psichiatrico)

Osserveremo ora il codice implementativo per ognuno dei trigger elencati precedentemente:

1

Creazione del Trigger per impedire a personali medici che non siano Psichiatri di effettuare delle Prescrizioni di trattamenti.

```
CREATE OR REPLACE TRIGGER Psichiatri_fanno_prescrizioni
BEFORE INSERT OR UPDATE ON Prescrizione
FOR EACH ROW

DECLARE

   discriminante_psichiatra NUMBER;
   non_e_uno_psichiatra EXCEPTION;

BEGIN

   SELECT Discriminante INTO discriminante_psichiatra
   FROM Personale_Medico
   WHERE Codice_albo = :NEW.Codice_albo_prescrizione;

   IF discriminante_psichiatra <> 1
   THEN RAISE non_e_uno_psichiatra;
   END IF;

EXCEPTION

   WHEN non_e_uno_psichiatra
   THEN RAISE_APPLICATION_ERROR(-20000, 'ERRORE! IL CODICE ALBO NON APPARTIENE AD UNO
PSICHIATRA!');

END;
```

2

Creazione del Trigger per impedire a personali medici che non siano Infermieri di effettuare delle Somministrazioni di trattamenti farmacologici.

```
CREATE OR REPLACE TRIGGER Infermieri_fanno_somministrazioni
BEFORE INSERT OR UPDATE ON Somministra
FOR EACH ROW

DECLARE

   discriminante_infermiere NUMBER;
   non_e_infermiere EXCEPTION;

BEGIN

   SELECT Discriminante INTO discriminante_infermiere
   FROM Personale_Medico
   WHERE Codice_albo = :NEW.Codice_albo_somm;

   IF discriminante_infermiere <> 2
   THEN RAISE non_e_infermiere;
   END IF;

EXCEPTION

   WHEN non_e_infermiere
   THEN RAISE_APPLICATION_ERROR(-20001, 'ERRORE! IL CODICE ALBO NON APPARTIENE AD UN
INFERMIERE!');

END;
```

3

Creazione del Trigger per impedire l'inserimento di pazienti in infermeria Se il loro Check Up è risultato Positivo, e stanno quindi bene.

```
CREATE OR REPLACE TRIGGER Il_ricovero_e_solo_per_malati
BEFORE INSERT OR UPDATE ON Ricovero
FOR EACH ROW

DECLARE

   risultato_CK Check_UP.risultato % TYPE;
   sta_bene EXCEPTION;

BEGIN

   SELECT risultato INTO risultato_CK
   FROM Check_Up
   WHERE ID_CheckUp = :NEW.ricovero_checkup;

   IF risultato_CK = 'P'
   THEN RAISE sta_bene;
   END IF;

EXCEPTION

   WHEN sta_bene
   THEN RAISE_APPLICATION_ERROR(-20002, 'ERRORE! IL CHECK-UP CHE STAI INSERENDO HA AVUTO ESITO
POSITIVO!');

END;
```

4

Creazione del trigger che impedisce di inserire nella tabella Effettua_CK personali medici che non siano Medici.

```
CREATE OR REPLACE TRIGGER Medici_fanno_CK
BEFORE INSERT OR UPDATE ON Effettua_ck
FOR EACH ROW

DECLARE

   discriminante_medico Personale_Medico.Codice_albo % TYPE;
   non_e_medico EXCEPTION;

BEGIN

   SELECT Discriminante INTO discriminante_medico
   FROM Personale_Medico
   WHERE Codice_albo = :NEW.Codice_albo_ck;

   IF discriminante_medico <> 4
   THEN RAISE non_e_medico;
   END IF;

EXCEPTION

   WHEN non_e_medico
   THEN RAISE_APPLICATION_ERROR(-20003, 'ERRORE! SOLO I MEDICI POSSONO EFFETTUARE CEHCK-UP');

END;
```

5

Creazione del trigger per impedire che persone diverse da psichiatri tengano le sedute.

```
CREATE OR REPLACE TRIGGER Psicoterapeuta_fanno_sedute
BEFORE INSERT OR UPDATE ON Seduta
FOR EACH ROW

DECLARE

   discr_psicoterapeuta_seduta Personale_Medico.Codice_albo % TYPE;
   non_e_psicoterapeuta EXCEPTION;

BEGIN

   SELECT Discriminante INTO discr_psicoterapeuta_seduta
   FROM Personale_Medico
   WHERE Codice_albo = :NEW.Codice_albo_s;

   IF discr_psicoterapeuta_seduta <> 3
   THEN RAISE non_e_psicoterapeuta;
   END IF;

EXCEPTION

   WHEN non_e_psicoterapeuta
   THEN RAISE_APPLICATION_ERROR(-20004, 'ERRORE! SOLO GLI PSICOTERAPEUTI POSSONO TENERE SEDUTE
PSICHIATRICHE!');

END;
```

6

Creazione del Trigger oer impedire che la Seduta si tenga nelle camere adibite all'accoglienza dei pazienti.

```
CREATE OR REPLACE TRIGGER stanza_adibita_a_sedute
BEFORE INSERT OR UPDATE ON Seduta
FOR EACH ROW

DECLARE

   discriminante_stanza Stanza.Discriminante_s % TYPE;
   stanza_non_adibita EXCEPTION;

BEGIN

   SELECT Discriminante_s INTO discriminante_stanza
   FROM Stanza
   WHERE Numero_stanza = :NEW.Numero_stanza_s;

   IF discriminante_stanza <> 2
   THEN RAISE stanza_non_adibita;
   END IF;

EXCEPTION

   WHEN stanza_non_adibita
   THEN RAISE_APPLICATION_ERROR(-20005, 'ERRORE! LA STANZA NON É ADIBITA ALLE SEDUTE!');

END;
```

7

Creazione del Trigger per impedire che ad un paziente venga associata una sala per le sedute psichiatriche invece che una camera.

```
CREATE OR REPLACE TRIGGER I_pazienti_stanno_nelle_camere
BEFORE INSERT OR UPDATE ON Paziente
FOR EACH ROW

DECLARE

   discriminante_camera Stanza.Discriminante_s % TYPE;
   non_e_una_stanza EXCEPTION;

BEGIN

   SELECT Discriminante_s INTO discriminante_camera
   FROM Stanza
   WHERE Numero_stanza = :NEW.Numero_stanza_paz;

   IF discriminante_camera <> 1
   THEN RAISE non_e_una_stanza;
   END IF;

EXCEPTION

   WHEN non_e_una_stanza
   THEN RAISE_APPLICATION_ERROR(-20006, 'ERRORE! I PAZIENTI POSSONO ESSERE INSERITI
ESCLUSIVAMENTE NELLE CAMERE, NON NELLE SALE!');

END;
```

8

Creazione del Trigger per impedire che nessuno a parte gli Psichiatri svolgano delle diagnosi ai pazienti.

```
CREATE OR REPLACE TRIGGER psichiatri_digniosticano
BEFORE INSERT OR UPDATE ON Diagnostica
FOR EACH ROW

DECLARE

   discrminante_psichiatra Personale_Medico.Discriminante % TYPE;
   diverso_da_psichiatra EXCEPTION;

BEGIN

   SELECT Discriminante INTO discrminante_psichiatra
   FROM Personale_Medico
   WHERE Codice_albo = :NEW.Codice_albo_dia;

   IF discrminante_psichiatra <> 1
   THEN RAISE diverso_da_psichiatra;
   END IF;

EXCEPTION

   WHEN diverso_da_psichiatra
   THEN RAISE_APPLICATION_ERROR(-20007, 'ERRORE! SOLO GLI PSICHIATRI FANNO UNA DIAGNOSTICA!');

END;
```

9

Creazione del trigger per impedire che vengano assegnati turni di pulizia a CUSTODI o SEGRATARI.

```
CREATE OR REPLACE TRIGGER Solo_gli_inservienti_puliscono
BEFORE INSERT OR UPDATE ON Turno_in
FOR EACH ROW

DECLARE

   discriminante_inserviente Personale_di_servizio.discriminante_pers_serv % TYPE;
   non_e_inserviente EXCEPTION;

BEGIN

   SELECT discriminante_pers_serv INTO discriminante_inserviente
   FROM Personale_di_servizio
   WHERE CF_Pers_serv = :NEW.CF_inserviente_turno;

   IF discriminante_inserviente <> 2
   THEN RAISE non_e_inserviente;
   END IF;

EXCEPTION

   WHEN non_e_inserviente
   THEN RAISE_APPLICATION_ERROR(-20008, 'ERRORE! I CUSTODI O I SEGRETARI PULISCONO!');

END;
```

10

Creazione del Trigger per impedire che INSERVIENTI o SEGRETARI custodiscano i reparti.

```
CREATE OR REPLACE TRIGGER I_custodi_custodiscono
BEFORE INSERT OR UPDATE ON Lavora_in
FOR EACH ROW

DECLARE

   discriminante_custode Personale_di_servizio.discriminante_pers_serv % TYPE;
   non_e_custode EXCEPTION;

BEGIN

   SELECT discriminante_pers_serv INTO discriminante_custode
   FROM Personale_di_servizio
   WHERE CF_Pers_serv = :NEW.CF_custode_turno;

   IF discriminante_custode <> 3
   THEN RAISE non_e_custode;
   END IF;

EXCEPTION

   WHEN non_e_custode
   THEN RAISE_APPLICATION_ERROR(-20009, 'ERRORE! UN INSERVIENTE OD UN SEGRETARIO NON SANNO
COME SI CHIUIDONO LE PORTE!');

END;
```

11

Creazione del Trigger che permette di associare nella tabella dei turni del personale medico, ad ogni individuo il suo reparto adeguato.

Esempio, non ha senso che un medico faccia un turno in un reparto "Mensa" il Trigger serve apposta per impedire che ciò accada.

```
CREATE OR REPLACE TRIGGER Ad_ognuno_il_suo_reparto
BEFORE INSERT OR UPDATE ON Fa_Turno
FOR EACH ROW

DECLARE

   dis_infermieristico Personale_Medico.Discriminante % TYPE;
   dis_psicoterapia Personale_Medico.Discriminante % TYPE;
   dis_camere Personale_Medico.Discriminante % TYPE;



BEGIN

   IF :NEW.codice_rep_turno = 1
   THEN RAISE_APPLICATION_ERROR(-20010, 'ERRORE! IL PERSONALE MEDICO NON FA TURNI DI LAVORO
NELLA MENSA!');
   END IF;

   IF :NEW.codice_rep_turno = 2 THEN

      SELECT Discriminante INTO dis_infermieristico
      FROM Personale_Medico
      WHERE Codice_albo = :NEW.pers_med;

      IF dis_infermieristico NOT IN (2,4)
      THEN RAISE_APPLICATION_ERROR(-20011, 'ERRORE! PSICHIATRI O PSICOTERAPEUTI NON LAVORANO IN
INFERMIERIA!');
      END IF;

   END IF;

   IF :NEW.codice_rep_turno = 3 THEN
```

```
    SELECT Discriminante INTO dis_psicoterapia
    FROM Personale_Medico
    WHERE Codice_albo = :NEW.pers_med;

    IF dis_psicoterapia NOT IN(1,3)
    THEN RAISE_APPLICATION_ERROR(-20012, 'ERRORE! SOLO PSICOTERAPEUTI E PSICHIATRI LAVORANO
NEL REAPRTO PSICHIATRIA!');
    END IF;

  END IF;

  IF :NEW.codice_rep_turno = 4 THEN

    SELECT Discriminante INTO dis_camere
    FROM Personale_Medico
    WHERE Codice_albo = :NEW.pers_med;

    IF dis_camere <> 2
    THEN RAISE_APPLICATION_ERROR(-20013, 'ERRORE! GLI INFERMIERI SONO GLI UNICI CHE HANNO
ACCESSO AL REPARTO CAMERE!');
    END IF;

  END IF;

END;
```

12

Creazione del Trigger per far sì che non si possano inserire valori nella colonna specializzazione se il discriminante è diverso da 4, ovvero se non sono medici.

```
CREATE OR REPLACE TRIGGER solo_medici_hanno_specializzazione
BEFORE INSERT OR UPDATE ON Personale_Medico
FOR EACH ROW

DECLARE

        spec_solo_per_medici EXCEPTION;

BEGIN

        IF :NEW.Discriminante <> 4 AND :NEW.Specializzazione IS NOT NULL
        THEN RAISE spec_solo_per_medici;
        END IF;

EXCEPTION

        WHEN spec_solo_per_medici
        THEN RAISE_APPLICATION_ERROR(-20014,'ERRORE! SOLO I MEDICI POSSONO AVERE UNA
SPECIALIZZAZIONE!');

END;
```

13

Creazione del trigger per impedire che un Check-Up positivo abbia indice di gravità diverso da 0

Non ha senso che un Paziente che sta bene, abbia un indice di gravità.

```
CREATE OR REPLACE TRIGGER Se_sta_bene_sta_bene
BEFORE INSERT OR UPDATE ON Check_Up
FOR EACH ROW

DECLARE

        sta_bene_ma_non_sta_bene EXCEPTION;

BEGIN

        IF :NEW.risultato = 'P' AND :NEW.Ind_grav <> 0
        THEN RAISE sta_bene_ma_non_sta_bene;
        END IF;

EXCEPTION

        WHEN sta_bene_ma_non_sta_bene
        THEN RAISE_APPLICATION_ERROR(-200015,'ERRORE! IL PAZIENTE NON PUÒ STARE BENE E MALE
ALLO STESSO TEMPO!');

END;
```

14

Creazione del Trigger per impedire che una Prenotazione risultata Rifiutata venga inserita nell'elenco Visite.

```
CREATE OR REPLACE TRIGGER No_visite_falze
BEFORE INSERT OR UPDATE ON Visita
FOR EACH ROW

DECLARE

        esito_prenotazione Prenotazione.esito % TYPE;
        prenotazione_rifiutata EXCEPTION;

BEGIN

        SELECT esito INTO esito_prenotazione
        FROM Prenotazione
        WHERE data_ora_visita = :NEW.data_visita_prenotazione AND CF_dipendenza = :NEW.CF_prenotazione_visita;

        IF esito_prenotazione <> 'Accettata'
        THEN RAISE prenotazione_rifiutata;
        END IF;

EXCEPTION

        WHEN prenotazione_rifiutata
        THEN RAISE_APPLICATION_ERROR(-20016, 'ERRORE! UN VISITATORE NON PUÒ EFFETTUARE UNA
VISITA SE QUESTA GLI È STATA RIFIUTATA!');

END;
```

15

Creazione del Trigger per impedire a pazienti minorenni di entrare in struttura.

```
CREATE OR REPLACE TRIGGER Paziente_Troppo_piccolo
BEFORE INSERT OR UPDATE ON Paziente
FOR EACH ROW

DECLARE

   troppo_piccolo EXCEPTION;
            eta_ingresso NUMBER;

BEGIN

   eta_ingresso := TRUNC(MONTHS_BETWEEN(:NEW.Data_ingresso,:NEW.DDN) / 12);
   IF eta_ingresso<18
   THEN RAISE troppo_piccolo;
   END IF;

EXCEPTION

   WHEN troppo_piccolo
   THEN RAISE_APPLICATION_ERROR(-20017, 'ERRORE! IL PAZIENTE DEVE AVERE ALMENO 18 ANNI!');

END;
```

16

Creazione del trigger per definire bene al momento della loro creazione le Stanze in modo che siano o sale o Camere.

```
CREATE OR REPLACE TRIGGER camere_so_camere_sale_so_sale
BEFORE INSERT OR UPDATE ON Stanza
FOR EACH ROW

DECLARE

        controllino_camerone EXCEPTION;
        controllino_stanzone EXCEPTION;

BEGIN

        IF :NEW.Discriminante_s = 1 THEN

                IF :NEW.codice_rep_cam <> 4 OR :NEW.codice_rep_s IS NOT NULL
                THEN RAISE controllino_camerone;
                END IF ;

        END IF;

        IF :NEW.Discriminante_s = 2 THEN

                IF :NEW.codice_rep_cam IS NOT NULL OR :NEW.codice_rep_s <> 3
                THEN RAISE controllino_stanzone;
                END IF ;

   END IF;

END;
```

Implementazione PL/SQL per le procedure degli utenti:

In questo paragrafo vengono osservate le 6 procedure degli utenti implementate nel database. Queste vengono commentate di pari passo, iniziamo ad osservarle:

- **Ricovero in infermeria:**

```
-- Creazione della procedura per la quale ad un check up negativo, viene effettuato
-- il ricovero del paziente e registrato il nome del medico che ha fatto il check up

CREATE OR REPLACE PROCEDURE Ricovero_in_infermieria (Codice_CK CHAR)
IS

  --Dichiariamo le variabili per conservare esito, gravitá e durata in giorni del ricovero
  esito_ck Check_Up.risultato % TYPE;
  gravita_ck Check_Up.Ind_grav % TYPE;
  durata_ricovero NUMBER;

  --Poi delle variabili per contenere nome e cognome del medico
  Nome_m Personale_Medico.Nome % TYPE;
  Cognome_m Personale_Medico.Cognome % TYPE;

  --Ed ancora le stesse per il paziente
  Nome_p Paziente.Nome % TYPE;
  Cognome_p Paziente.Cognome % TYPE;

BEGIN

  --Selezioniamo l'esito tramite il Codice_CK preso come argomento dalla tabella CHECK_UP
  SELECT risultato INTO esito_ck
  FROM Check_Up
  WHERE ID_CheckUp = Codice_CK;

  --Se l'esito é negativo avviene la procedura
  IF esito_ck <> 'Positivo' THEN
```

```
--Prendiamo l'indice di gravitá che ci aiuta a scegliere i giorni di ricovero
SELECT Ind_grav INTO gravita_ck
FROM Check_Up
WHERE ID_CheckUp = Codice_CK;

--Scegliamo i giorni di ricovero
CASE gravita_ck
   WHEN 1 THEN durata_ricovero := 1;
   WHEN 2 THEN durata_ricovero := 2;
   WHEN 3 THEN durata_ricovero := 5;
   WHEN 4 THEN durata_ricovero := 10;
   WHEN 5 THEN durata_ricovero := 30;
                    ELSE durata_ricovero := 60;
END CASE;

--Inseriamo nella tabella Ricovero il Check_UP a cui corrisponde il paziente da ricoverarte
INSERT INTO Ricovero (giorni_durata, ricovero_inferm, ricovero_checkup) VALUES (durata_ricovero, 2,
Codice_CK);

--Prendiamo ora nome e congome di Medico e Paziente Tramite queste Query innestate
--L'innesto è implementato per trovare il codice fiscale del paziente dalla tabella Check_up
--alla quale corrisponde il Codice_CK argomento
--Lo stesso principio vale per nome e cognome del medico, nell'innesto troviamo una condizione
--di join poiché il codice albo di chi ha effettuato il Check si trova sulla tabella Effettua_CK
SELECT Nome, Cognome INTO Nome_m, Cognome_m
FROM Personale_Medico
WHERE Codice_albo = ( SELECT Codice_albo_ck
                FROM (Check_UP c join Effettua_CK ec ON c.ID_CheckUp = ec.ck_up_eff)
                WHERE ID_CheckUp = Codice_CK );

SELECT Nome, Cognome INTO Nome_p, Cognome_p
FROM Paziente
WHERE CF = ( SELECT CF_checkup
          FROM Check_Up
```

```
        WHERE ID_CheckUp = Codice_CK);


    --Stampiamo ora un messaggio per riassumere cosa è successo

     DBMS_OUTPUT.PUT_LINE('Al paziente '|| Nome_p || ' ' || Cognome_p || ' sono stati assegnati '|| durata_ricovero || ' giorni di ricovero dal Dott. '|| Nome_m || ' ' || Cognome_m||'.');


  --In caso il risultato di quel CHECK_UP sia positivo, non vi é motivo di effettuare

  --nulla e viene generato pertanto un messaggio di errore

  ELSE


     RAISE_APPLICATION_ERROR(-20040, 'ERRORE! NON ESISTE ALCUN CHECK-UP NEGATIVO CON QUEL CODICE!');


  END IF;


END;
```

- **Prenotazione accettata:**

```
--Creazione della procedura che inserisce una prenotazione accettata nella tabella visita
CREATE OR REPLACE PROCEDURE Prenotazione_accettata(codice_visitatore CHAR, data_e_ora TIMESTAMP )
IS

   --Dichiariamo le variabili che useremo durante la procedura
   risultato Prenotazione.esito % TYPE;
   CF_visitato Prenotazione.CF_paziente % TYPE;

   Nome_p Paziente.Nome % TYPE;
   Cognome_p Paziente.Cognome % TYPE;

   Nome_v Visitatore.nome_visitatore % TYPE;
   Cognome_v  Visitatore.cog_visitatore % TYPE;

BEGIN

   --Iniziamo selezionando l'esito ed il codice fiscale di chi deve
   --Essere visitato dalla Prenotazione che prendiamo in considerazione
   SELECT esito, CF_paziente INTO risultato, CF_visitato
   FROM Prenotazione
   WHERE CF_dipendenza = codice_visitatore AND data_ora_visita = data_e_ora;

   --Se l'esito è Accettata allora viene eseguito il comando per l'inserimento
   --della porenotazione nella tabella Visite
   IF risultato = 'Accettata' THEN

      INSERT INTO Visita(data_visita_prenotazione, CF_prenotazione_visita, CF_visita_paziente) VALUES
(data_e_ora, codice_visitatore, CF_visitato );

      --Quello che vogliamo ora è, un po' come nella procedura precedente
      --Scrivere a schermo un messggio che confermi la prenotazione
      --Pertanto sempre tramite query innestate andiamo a selezionare
      --Nome e cognome del visitatore e del visitato da usare come parametri
```

```
--Per il messaggio di output
SELECT Nome, Cognome INTO Nome_p, Cognome_p
FROM Paziente
WHERE CF = ( SELECT CF_paziente
        FROM Prenotazione
        WHERE data_ora_visita = data_e_ora AND CF_dipendenza = codice_visitatore );

SELECT nome_visitatore, cog_visitatore INTO Nome_v, Cognome_v
FROM Visitatore
WHERE CF_visitatore = ( SELECT CF_dipendenza
            FROM Prenotazione
            WHERE data_ora_visita = data_e_ora AND CF_dipendenza = codice_visitatore );

--Scriviamo il messaggio
DBMS_OUTPUT.PUT_LINE( Nome_v||' '||Cognome_v|| ' visiterà ' || Nome_p || ' ' || Cognome_p || ' in data ' || data_e_ora || '.');

--In caso la prenotazione sia stata rifiutata viene generato un messaggio di errore
ELSE

RAISE_APPLICATION_ERROR(-20041, 'ERRORE! LA PRENOTAZIONE INSERITA NON É STATA ACCETTATA!');

END IF;

END;
```

- **Resoconto della giornata:**

```
--Creazione di una procedura che quando eseguita mostra tutto ciò che un
--Paziente ha fatto in quella giornata
CREATE OR REPLACE PROCEDURE Resoconto_giornata (codice_paziente CHAR, data_scelta DATE)
IS

   nome_paziente Paziente.Nome % TYPE;
   cognome_paziente Paziente.Cognome % TYPE;
   nome_tratt Somministra.Nome_tratt_somm % TYPE;
   esito Check_Up.risultato % TYPE;

   --Questa variabile booleana serve a tenere traccia dell'attivazione
   --NO_DATA_FOUND, se questa non viene attivata, allora sono stati trovati dati
   --e possono venire comunicati, se si attiva, viene scritto che il paziente
   --selezionato non ha ricevuto somm in quella data
   flag_somm BOOLEAN := FALSE;
   flag_seduta BOOLEAN := FALSE;
   flag_Check_up BOOLEAN := FALSE;

BEGIN

   --Query innestata per ottenere il Nome ed il cognome del paziente se ha
   --Subito una certa somministrazione di un trattmaneto

   --Nella query  viene eseguito un join tra un'altra query che restituisce una sola tupla
   --e la tabella paziente che contiene il nome, referenziato tramite chiave esterna

   --Osservazione importante é che se la query restituisce 0 valori,
   --verrá gestito l'errore no data found. Questa eccezione viene implementata
   --come nuovo blocco BEGIN END;
   BEGIN

      SELECT Nome, Cognome, Nome_tratt_somm INTO nome_paziente, cognome_paziente, nome_tratt
```

```
    FROM (( SELECT Nome_tratt_somm, CF_somm
        FROM Somministra
        WHERE data_ora_somm IN ( TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 09:00:00', 'DD-MM-YYYY HH24:MI:SS'),
                                 TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 15:00:00', 'DD-MM-YYYY HH24:MI:SS'),
                                 TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 21:00:00', 'DD-MM-YYYY HH24:MI:SS'),
                                 TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 09:30:00', 'DD-MM-YYYY HH24:MI:SS'))
                                 AND CF_somm = codice_paziente
                                 GROUP BY Nome_tratt_somm, CF_somm) a join Paziente on a.CF_somm = Paziente.CF);

    EXCEPTION

      --Se la queri non restituisce valori viene gestito l'essore
       WHEN NO_DATA_FOUND THEN
       flag_somm := TRUE;
      DBMS_OUTPUT.PUT_LINE('Il paziente selezionato NON ha subito trattamenti il giorno '|| TO_CHAR(data_scelta, 'DD-MM-YYYY'));

  END;

  --Se la query ha restituito valori questi vengono comunicati
  IF NOT flag_somm THEN

    DBMS_OUTPUT.PUT_LINE('Paziente: ' || nome_paziente || ' ' || cognome_paziente);
    DBMS_OUTPUT.PUT_LINE('Trattamento somministrato: ' || nome_tratt);

  END IF;

  --Lo stesso raginamento precedente viene applcato anche alle sedute ed ai chech-Up

  --Blocco per le Sedute
  BEGIN

    --Selzioniamo nome e cognome tramite il codice fiscale
```

```
SELECT Nome, Cognome INTO nome_paziente, cognome_paziente
FROM Paziente
WHERE CF = codice_paziente;

--Svolgiamo la query per ottenere il Nome del trattamento associato al nome del paziente
SELECT Nome, Cognome, nome_tratt_s INTO nome_paziente, cognome_paziente, nome_tratt
FROM ( (SELECT Nome_tratt_s, CF_s
    FROM Seduta
    WHERE Data_ora_s IN ( TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 10:30:00', 'DD-MM-YYYY HH24:MI:SS'),
                TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 11:30:00', 'DD-MM-YYYY HH24:MI:SS'),
                TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 12:30:00', 'DD-MM-YYYY HH24:MI:SS'),
                TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 13:30:00', 'DD-MM-YYYY HH24:MI:SS'),
                TO_DATE(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 14:30:00', 'DD-MM-YYYY HH24:MI:SS'))

    GROUP BY Nome_tratt_s, CF_s ) s JOIN Paziente p ON s.CF_s = p.CF)
WHERE Nome = nome_paziente AND Cognome = cognome_paziente;

--Gestiamo l'eccezione nel caso la query di cui sopra non generi risultati
EXCEPTION

  WHEN NO_DATA_FOUND THEN

    flag_seduta := TRUE;
    DBMS_OUTPUT.PUT_LINE('Il paziente selezionato NON ha effettuato SEDUTE il giorno ' || TO_CHAR(data_scelta, 'DD-MM-YYYY'));

END;

--Nel caso vi siano dei risultati vengono descritti
IF NOT flag_seduta THEN

  DBMS_OUTPUT.PUT_LINE('Ha effettuato una SEDUTA di: '|| nome_tratt);
```

```
END IF;

--Blocco per i Check-Up
BEGIN

    SELECT Nome, Cognome, Risultato INTO nome_paziente, cognome_paziente, esito
    FROM (( SELECT *
        FROM Check_Up c JOIN Effettua_ck ec ON c.ID_checkup = ec.ck_up_eff) a JOIN Paziente p ON p.CF =
a.CF_checkup)

    --Come condizione di where vengono impostate date limite per un singolo giorno,
    --quello selezionato, dato che in un giorno viene effettuato un solo check-up per paziente
    WHERE    Data_ora_ck >= TO_TIMESTAMP(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 09:00:00', 'DD-
MM-YYYY HH24:MI:SS')
        AND Data_ora_ck <= TO_TIMESTAMP(TO_CHAR(data_scelta, 'DD-MM-YYYY') || ' 23:59:59', 'DD-MM-
YYYY HH24:MI:SS')
        AND CF = codice_paziente;

    EXCEPTION

      WHEN NO_DATA_FOUND THEN

        flag_Check_up := TRUE;
        DBMS_OUTPUT.PUT_LINE('Il paziente selezionato NON ha effettuato CHECK-UP il giorno ' ||
TO_CHAR(data_scelta, 'DD-MM-YYYY'));

END;

IF NOT flag_Check_up THEN

 DBMS_OUTPUT.PUT_LINE('Ha effettuato una CHECK-UP risultato: '|| esito || '.' );

END IF;

END;
```

- **Medicinali del paziente:**

```
--Creazione della procedura che ci dice ogni paziente che subisce un trattamento
--farmacologico, quale medicinale usa. Viene implementata come procedura poiché è
--query poco intuitiva e molto lunga
CREATE OR REPLACE PROCEDURE Medicinali_del_paziente (codice_fiscale CHAR)
IS

   nome_paziente Paziente.Nome%TYPE;
   cognome_paziente Paziente.Cognome%TYPE;
   nome_med Medicinale.Nome_med%TYPE;

BEGIN

   SELECT Nome, Cognome INTO nome_paziente, cognome_paziente
   FROM Paziente
   WHERE CF = codice_fiscale;

   DBMS_OUTPUT.PUT_LINE('Il paziente ' || nome_paziente || ' ' || cognome_paziente || ' prende:');

   FOR med_rec IN (

      SELECT Nome_Med
      FROM ((
            SELECT CF, Nome, Cognome, ATC_usato
            FROM (
                SELECT CF, Nome, Cognome, Nome_tratt_fatto
                FROM (Paziente p JOIN Ha_tratt t ON p.CF = t.CF_paziente_farm)
               ) a JOIN Fa_uso f ON a.Nome_tratt_fatto = f.Nome_tratt_uso ) b JOIN Medicinale m ON b.ATC_usato =
m.ATC )

      WHERE CF = codice_fiscale

   ) LOOP
```

```
        nome_med := med_rec.Nome_Med;
        DBMS_OUTPUT.PUT_LINE('Medicinale: ' || nome_med);

    END LOOP;
END;
```

- **Lavoratori nel reparto:**

```
--Creazione della procedura che data una data ed un numero reparto, mostra le guardie ed il personale
--di servizio che lavora all'interno di quel reparto
CREATE OR REPLACE PROCEDURE Lavoratori_nel_reparto(nro_rep NUMBER, data_lav DATE)
IS

   nome Guardia.Nome_g % TYPE;
        cognome Guardia.Cognome_g % TYPE;

        nome_rep Reparto.nome_reparto % TYPE;

BEGIN

        SELECT Nome_reparto INTO nome_rep
   FROM REPARTO
   WHERE codice_rep = nro_rep;

        DBMS_OUTPUT.PUT_LINE('Nel reparto '|| nome_rep || ' nel giorno '|| data_lav || 'lavorano:');

   --For che cicla tutte le tuple della query che trova le guardie che lavorano
   --nel reparto selezionato nella data selezionata
   FOR tupla_guardia IN (

      Select Nome_g, Cognome_g
      FROM (( Select *
                  From Effettua_turno
                  WHERE data_turno_g = data_lav AND cod_rep_turno_g = nro_rep) a
           JOIN Guardia g ON a.matricola_turno = g.Matricola)

   ) LOOP

        nome := tupla_guardia.nome_g;
```

```
        cognome := tupla_guardia.cognome_g;

    --Stampa della persona appena selezionata dal FOR
        DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Guardia');

END LOOP;

--For che cicla tutte le tuple della query che trova gli inservienti
--che lavorano nel reparto selezionato nella data selezionata
FOR tupla_inserviente IN (

    SELECT Nome_dip, Cognome_dip
    FROM (( SELECT *
          FROM Turno_in
          WHERE codice_rep_inser = nro_rep AND data_turno = data_lav) a
          JOIN Personale_di_servizio p on a.CF_inserviente_turno = p.CF_Pers_serv)

) LOOP


    nome := tupla_inserviente.Nome_dip;
    cognome := tupla_inserviente.Cognome_dip;

    --Stampa della persona appena selezionata dal FOR
    DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Inserviente');


END LOOP;

--For che cicla tutte le tuple della query che trova i custodi
--che lavorano nel reparto selezionato nella data selezionata

        FOR tupla_custode IN(
```

```
    SELECT Nome_dip, Cognome_dip
    FROM (( Select *
          FROM Lavora_in
          WHERE data_turno = data_lav AND codice_rep_custode = nro_rep) a
          JOIN Personale_di_servizio p ON a.CF_custode_turno = p.CF_Pers_serv)

  ) LOOP

    nome := tupla_custode.Nome_dip;
    cognome := tupla_custode.Cognome_dip;

    --Stampa della persona appena selezionata dal FOR
    DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Custode');

          END LOOP;

END;
```

- **Turni Personale Medico:**

```
-- Creazione della procedura che mostra in quale reparto lavora ogni persona del personale medico
-- Viene implementata come procedura separata dalla precedente poiché ricadono Specializzazioni
-- diverse per ognuno del personale medico, in più coloro che ne fanno parte non lavorano in mensa
-- dobbiamo quindi gestire anche un'eccezione
CREATE OR REPLACE PROCEDURE Turni_personale_medico (nro_rep NUMBER, turno_pers DATE)
IS

   nome Personale_Medico.Nome % TYPE;
         cognome Personale_Medico.Cognome % TYPE;
   valore_discr Personale_Medico.Discriminante % TYPE;

         nome_rep Reparto.nome_reparto % TYPE;

   -- Creazione dell'eccezione nel caso si sia inserito un
   -- numero reparto sbagliato
   codice_rep_sbagliato EXCEPTION;

BEGIN

   -- Gestiamo un'eccezione nel caso il numero reparto inserito
   -- sia uguale ad uno. Il personale_medico non lavora in mensa,
   -- esiste anche un trigger apposito per gestire questo vincolo
   IF nro_rep = 1
   THEN
      RAISE codice_rep_sbagliato;
   END IF;

   -- Selezioniamo il nome del reparto e lo scriviamo
   SELECT Nome_reparto INTO nome_rep
   FROM REPARTO
   WHERE codice_rep = nro_rep;
```

```
        DBMS_OUTPUT.PUT_LINE('Nel reparto '|| nome_rep ||' nel giorno '|| turno_pers || ' lavorano:');


  -- Cicliamo sulla tabella del personale medico che lavora in quel reparto
  -- nell'ora indicata, per leggere tutte le tuple generate dalla query
  FOR tupla_pers_med IN (

    SELECT Nome, Cognome, Discriminante
    FROM ( (SELECT * FROM Fa_turno
          WHERE data_turno = turno_pers AND codice_rep_turno = nro_rep) a
          JOIN Personale_medico p ON a.Pers_med = p.Codice_albo)

  ) LOOP

    nome := tupla_pers_med.Nome;
    cognome := tupla_pers_med.Cognome;
    -- Conserviamo il discriminante oltre che il nome ed il cognome
    valore_discr := tupla_pers_med.Discriminante;

    -- In base al discriminante scriviamo un messaggio diverso
    CASE valore_discr
      WHEN 1 THEN DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Psichiatra');
      WHEN 2 THEN DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Infermiere');
      WHEN 3 THEN DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Psicoterapeuta');
      ELSE       DBMS_OUTPUT.PUT_LINE(nome || ' ' || cognome || ' --> Medico');
    END CASE;

  END LOOP;
-- Gestione dell'eccezione
EXCEPTION

  WHEN codice_rep_sbagliato THEN
    RAISE_APPLICATION_ERROR(-20045, 'ERRORE! HAI INSERITO UN CODICE REPARTO
SBAGLIATO!');

END;
```