

Out!

Contents

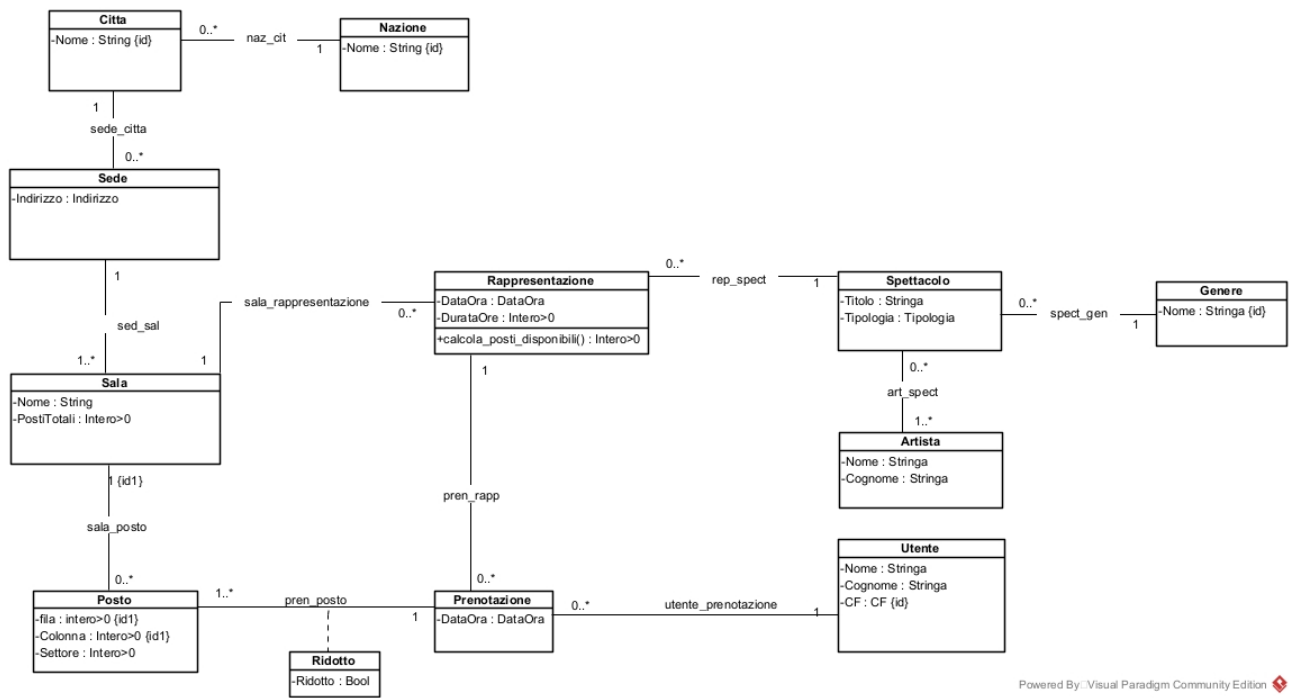
1	Requisiti	3
2	UML	4
3	Tipi di Dato	5
4	Vincoli Esterni	5
5	Specifica Classi	6
5.1	Classe Rappresentazione	6
5.1.1	calcola_posti_disponibili(): Intero	6
6	Use-Case	7
6.1	Diagramma	7
6.2	Specifica Use-Case	8
6.2.1	Iscrizione	8
6.2.2	Prenotazione	8
6.2.3	Consulta Lista	9
6.2.4	Suggerimenti	9
7	Ristrutturazione	10
7.1	Diagramma UML ristrutturato	10
7.2	Tipi e Domini	11
7.2.1	Tipi	11
7.2.2	Domini	11
7.3	Vincoli Esterni	11
7.4	Use Case	11
7.5	Traduzione diretta del diagramma UML delle classi ristrutturato	12
7.6	Trigger	13
7.6.1	V.Prenotazione.Data	13
7.6.2	V.Rappresentazione.Simultanea	13
7.6.3	V.Sala.Posti	14
7.6.4	V.Posto.PostiDiversi	14
7.6.5	V.Prenotazione.PostoInSala	15
7.7	Progettazione Funzionalità	16
7.7.1	calcola_posti_disponibili	16
7.7.2	iscrizione	16
7.7.3	prenotazione	17

7.7.4	consulta_lista	17
7.7.5	proponi_suggerimento	18

1 Requisiti

1. Utente
 - 1.1 Cf
 - 1.2 Nome
 - 1.3 Cognome
2. Prenotazione
 2. DataOra
3. Posto
 - 3.1 Fila
 - 3.2 Colonna
 - 3.3 Settore
4. Sala
 - 4.1 Nome
 - 4.2 PostiTotali
5. Sede
 - 5.1 Indirizzo
6. Citta
 - 6.1 Nome
7. Nazione
 - 7.1 Nome
8. Rappresentazione
 - 8.1 DataOra
 - 8.2 DurataOre
9. Spettacolo
 - 9.1 Titolo
 - 9.2 Tipologia
10. Genere
 - 10.1 Nome
11. Artista
 - 11.1 Nome
 - 11.2 Cognome

2 UML



Powered By: Visual Paradigm Community Edition

3 Tipi di Dato

- Tipologia= { "Film", "Rappresentazione Teatrale", "Concerto" }
- CF = Stringa secondo Regex $[A-Z]\{6\}[0-9]\{2\}[A-Z][0-9]\{2\}[A-Z][0-9]\{3\}[A-Z]$

4 Vincoli Esterni

- $[V.Prenotazione.Data]$

Non è possibile prenotare per uno spettacolo passato

$$\forall p, d, a, g \text{ Prenotazione}(p) \wedge \text{DataPrenotazione}(p, d) \wedge \text{Adesso}(a) \wedge \text{GiornoOdierno}(a, g) \rightarrow g > d$$

- $[V.Rappresentazione.Simultanea]$

Non più di 1 rappresentazione per sala nello stesso momento

$$\begin{aligned} &\forall r, s, g, h, d, r', s', g', h', d' \\ &\text{Rappresentazione}(r) \wedge \text{Sala}(s) \wedge \text{SalaRappresentazione}(r, s) \wedge \text{giornoRappresentazione}(r, g) \wedge \\ &\text{oraGiornoRapp}(r, g, h) \wedge \text{DurataRappresentazione}(r, d) \wedge \\ &\text{Rappresentazione}(r') \wedge \text{Sala}(s') \wedge \text{SalaRappresentazione}(r', s') \wedge \text{giornoRappresentazione}(r', g') \wedge \\ &\text{oraGiornoRapp}(r', g', h') \wedge \text{DurataRappresentazione}(r', d') \wedge \\ &r! = r' \wedge s = s' \wedge g = g' \rightarrow (h + d < h' \vee h > h' + d') \end{aligned}$$

- $[V.Sala.Posti]$

N PostiPrenotazione \leq Sala.PostiTotali

$$\begin{aligned} &\forall s, p \text{ Sala}(s) \text{ calcola_posti_liberi}(s, p) \\ &p > 0 \end{aligned}$$

- $[V.Posto.PostiDiversi]$

Non possono esserci due posti uguali nella stessa sala legati a due prenotazioni diverse

$$\forall ps \text{ Posto}(ps) \rightarrow \neg p1, p2, r \text{ Prenotazione}(p1) \wedge \text{Prenotazione}(p2) \wedge \text{pren_posto}(p1, ps) \wedge \text{pren_posto}(p2, ps) \wedge \text{pren_rap}(p1, r) \wedge \text{pren_rap}(p2, r) \wedge p1! = p2$$

- $[V.Prenotazione.PostoInSala]$

Se un utente prenota per una rappresentazione e un determinato posto esso si trova nella sala della rappresentazione

$$\forall p, r, ps, s \text{ Prenotazione}(p) \wedge \text{Rappresentazione}(r) \wedge \text{pren_rap}(p, r) \wedge \text{pren_posto}(p, ps) \wedge \text{SalaRappresentazione}(s, r) \rightarrow \text{sala_posto}(ps, s)$$

5 Specifica Classi

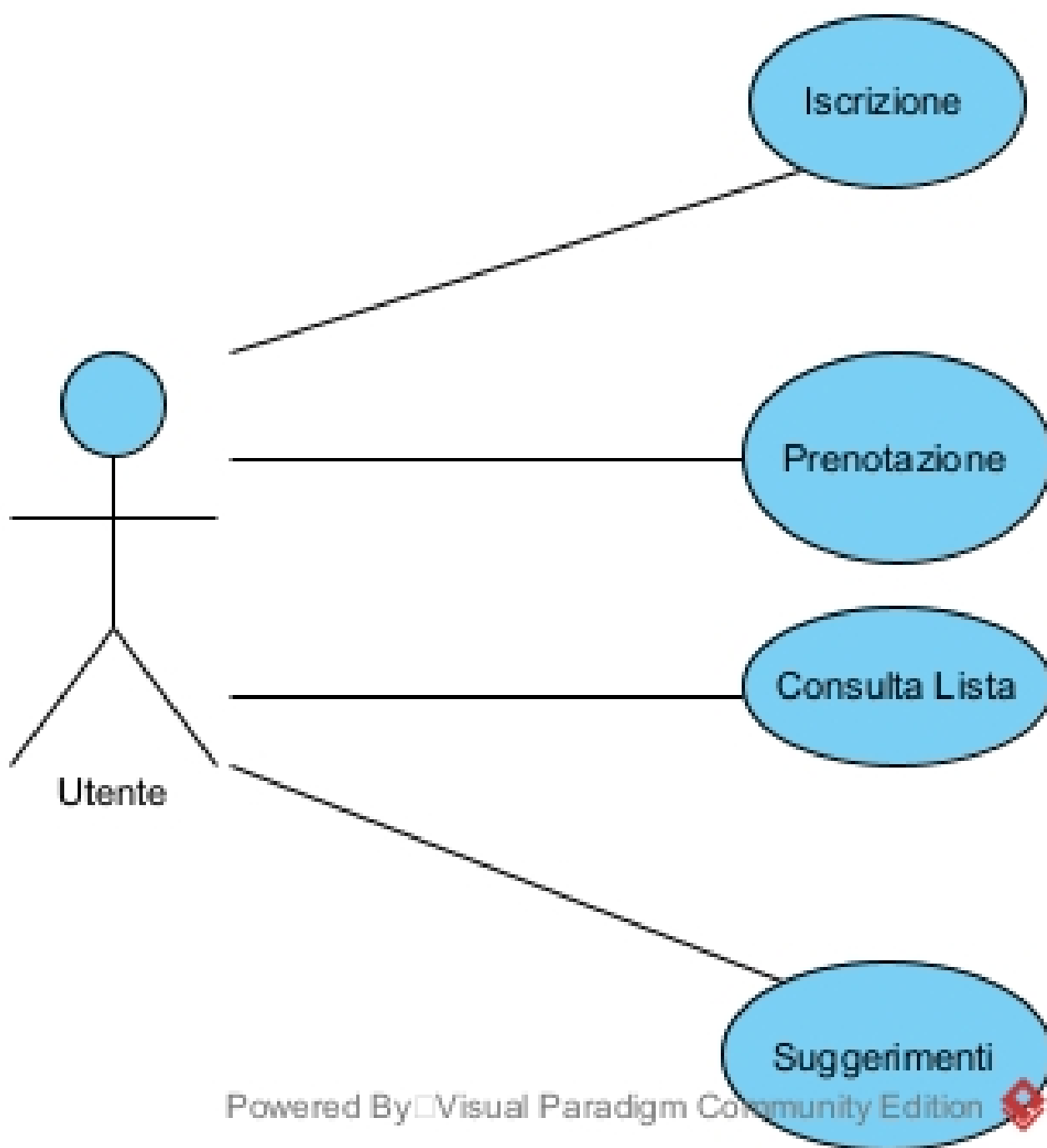
5.1 Classe Rappresentazione

5.1.1 `calcola_posti_disponibili()`: Intero

- PreCondizioni: nessuna
- Modifiche al livello estenzionale: nessuna
- PostCondizioni:
 $\forall s, p \text{ Sala}(s) \text{ PostiSala}(s, p)$
 $P = \{(p, s) \text{ Posto}(p) \wedge \text{Sala}(s, p)\}$
Result è uguale a $\text{PostiSala} - |P|$

6 Use-Case

6.1 Diagramma



6.2 Specifica Use-Case

6.2.1 Iscrizione

iscrizione(n: Stringa, c: Stringa, cf: CF): Utente
PreCondizioni: nessuna
PostCondizioni:
 Modifica il livello estensionale quindi $Min \neq Mout$.
 Elementi in D aggiunti: p
 Elementi in D rimossi: nessuno
 Oggetti aggiunti:
 -Utente(p)
 -nome(p,n)
 -cognome(p,c)
 -CF(p,cf)
 Oggetti rimossi: nessuno
 result = p

6.2.2 Prenotazione

prenotazione(u: Utente, r: Rappresentazione, n: Intero, s: Intero): Prenotazione

- PreCondizioni:
 $\exists Utente(u) \wedge Rappresentazione(r) \wedge SalaRappresentazione(s, r) \wedge calcola_posti_liberi(s, p) \geq n$.
- PostCondizioni:

Modifica il livello estensionale quindi $Min \neq Mout$.
Elementi in D aggiunti: p
Elementi in D rimossi: nessuno
Oggetti aggiunti:
 - Prenotazione(p)
 - pren_rap(p,r)
 - utente_prenotazione(p,u)
 - Sia Sala(s) tale che SalaRappresentazione(r,s)
 - Vengono creati n oggetti pren_post(p)

6.2.3 Consulta Lista

consulta_lista(t : Stringa, g : Stringa, d : Data) : Insieme

- PreCondizioni:

$\exists Rappresentazione(r) \wedge TipoRappresentazione(r, t) \wedge GenereRappresentazione(r, g) \wedge$
 $GiornoRapp(r, d)$

- PostCondizioni:

Non modifica il livello estensionale quindi Min=Mout.

Elementi in D aggiunti: nessuno

Elementi in D rimossi: nessuno

- Sia $R = \{r Rappresentazione(r) \wedge TipoRappresentazione(r, t) \wedge GenereRappresentazione(r, g) \wedge$
 $GiornoRapp(r, d)\}$ Result = R

6.2.4 Suggerimenti

proponi_suggerimento() : Insieme

- PreCondizioni: nessuna

- PostCondizioni:

Non modifica il livello estensionale quindi Min=Mout.

Elementi in D aggiunti: nessuno

Elementi in D rimossi: nessuno

- Sia Utente(this)

- Sia $R = \{Rappresentazione(r) | utente_pren(this, p) \wedge pren_rap(p, r)\}$

- Sia UltimaRappresentazione(r) la rappresentazione più recente in R

- Sia GenereRappresentazione(r, g)

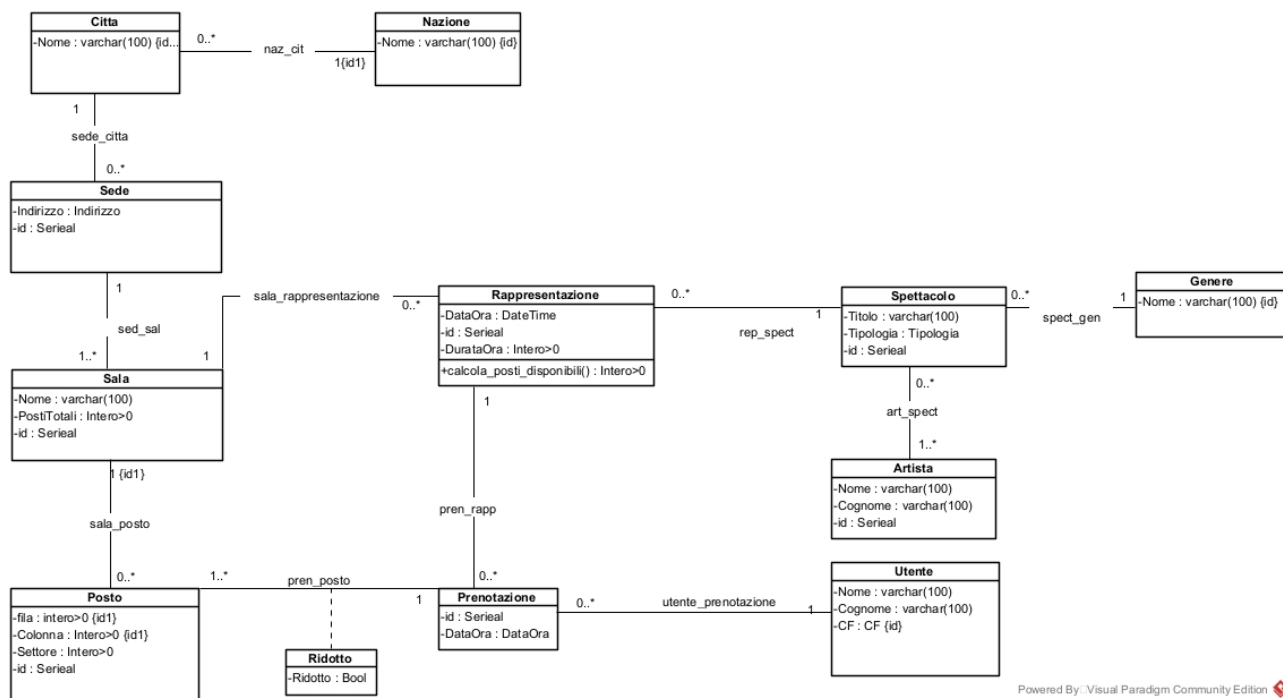
- Sia $S = \{Rappresentazione(r) | GenereRappresentazione(r, g) \wedge GiornoRapp(r, d)$
 $\wedge adesso(a) \wedge GiornoAttuale(ga) \wedge d > ga + 7\}$

Result = S

Arrivati a questo punto la fase di **Analisi** è **finita**.

7 Ristrutturazione

7.1 Diagramma UML ristrutturato



Powered By: Visual Paradigm Community Edition

Modifiche sulle generalizzazioni effettuate: Non è stato necessario eseguire nessuna modifica sulle generalizzazioni.

7.2 Tipi e Domini

7.2.1 Tipi

- create type *Tipologia* as enum= {*"Film"*,*"RappresentazioneTeatrale"*,*"Concerto"*}
- create type *Indirizzo*= {*via* : *string_not_null*,*civico* : *Intero* ≥ 0}

7.2.2 Domini

- create domain *intero* > 0 as Integer check(*value* ≥ 0)
- create domain CF as varchar check(*value* rispetta la regex $[A-Z]\{6\}[0-9]\{2\}[A-Z][0-9]\{2\}[A-Z][0-9]\{3\}[A-Z]$)
- create domain *string_not_null* as varchar check(*value* ≠ NULL)

7.3 Vincoli Esterni

Non sono stati inseriti nuovi vincoli esterni.

7.4 Use Case

Gli use case non violano la nuova ristrutturazione.

7.5 Traduzione diretta del diagramma UML delle classi ristrutturato

Saranno scritte tutte le tabelle da creare:

```
- Nazione(_nome_: varchar)

- Citta(_nome_: varchar, _nazione_: varchar)
  fk: nazione references Nazione(nome)

-Sede(_id_: Serial, Indirizzo: Indirizzo, _citta_: varchar)
  fk: citta references Citta(nome)

-Sala(_id_: Serial, Nome: varchar, PostiTotali: intero, Sede: Intero)
  fk; sede references Sede(id)

-Posto(_id_: Serial, Fila: Intero>0, Colonna: Intero>0,
       Settore: Intero>0, Sala: Intero)
  unique(Fila, Colonna, Settore, Sala)
  fk: sala references Sala(id)

-Rappresentazione(_id_: Serial, TDataOra: DateTime,
                  DurataOre: Intero>0, Sala: Intero, Spettacolo: intero>0)
  fk: sala references Sala(id)
  fk: Spettacolo references Spettacolo(id)

-Spettacolo(_id_: Serial, Titolo: varchar, Tipologia: Tipologia, Genere: varchar)
  fk: Genere references Genere(Nome)

-Genere(_nome_: varchar)

-Artista(_id_: Serial, Nome: varchar, Cognome: varchar)

-art_spect(_Spettacolo_: Intero, _Artista_: Intero)
  fk: Spettacolo references Spettacolo(id)
  fk: Artista references Artista(id)

-Utente(_CF_: CF, Nome: varchar, Cognome: varchar)

-Prenotazione(_id_: Serial, DataOra: DateTime, Utente:
              CF, Rappresentazione: Intero)
  fk: utente references Utente(CF)
  fk: rappresentazione references Rappresentazione(id)

- pren_posto(_Prenotazione_: Intero, _Posto_: Intero)
  fk: Prenotazione references Prenotazione(id)
  fk: Posto references Posto(id)
```

Sono state accorpate tutte le associazioni tranne *art_spect* e *pren_posto*.

7.6 Trigger

I vincoli esterni da controllare con i trigger sono:

[*V.Prenotazione.Data*]

[*V.Rappresentazione.Simultanea*]

[*V.Sala.Posti*]

[*V.Posto.PostiDiversi*]

[*V.Prenotazione.PostoInSala*]

7.6.1 V.Prenotazione.Data

Trigger per il vincolo V.Prenotazione.Data:

Operazioni: inserimento o modifica in Prenotazione

Istante di invocazione: prima dell'operazione intercettata

Funzione:

```
Sia isError=FALSE;
Sia new l'ennupla che si sta inserendo oppure
    l'ennupla risultato della modifica;
isError:= exists(select *
                  from new
                  where new.DataOra < adesso());
Se isError = TRUE blocca l'operazione;
Altrimenti permetti l'operazione
```

7.6.2 V.Rappresentazione.Simultanea

Trigger per il vincolo V.Rappresentazione.Simultanea:

Operazioni: inserimento o modifica in Rappresentazione

Istante di invocazione: prima dell'operazione intercettata

Funzione:

```
Sia isError=FALSE;
Sia new l'ennupla che si sta inserendo oppure
    l'ennupla risultato della modifica;
isError:= exists(select *
                  from Rappresentazione r
                  where new.Sala=r.Sala and new.DataOra=r.DataOra
                  and new.id!=r.id);
Se isError = TRUE blocca l'operazione;
Altrimenti permetti l'operazione
```

7.6.3 V.Sala.Posti

Trigger per il vincolo V.Sala.Posti:

Operazioni: inserimento o modifica in Prenotazione

Istante di invocazione: prima dell'operazione intercettata

Funzione:

```
Sia isError=FALSE;
Sia new l'ennupla che si sta inserendo oppure
    l'ennupla risultato della modifica;
isError:= exists(select *
                from Sala s, (select count(*)
                             from Posto p, pren_posto pp
                             where new.id= pp.id and p.Sala= s.id
                             where s.calcola_posti_liveri() < q)
                where s.calcola_posti_liveri() < q)
Se isError = TRUE blocca l'operazione;
Altrimenti permetti l'operazione
```

7.6.4 V.Posto.PostiDiversi

Trigger per il vincolo V.Posto.PostiDiversi:

Operazioni: inserimento o modifica in Posto

Istante di invocazione: prima dell'operazione intercettata

Funzione:

```
Sia isError=FALSE;
Sia new l'ennupla che si sta inserendo oppure
    l'ennupla risultato della modifica;
isError:= exists( select *
                from pren_posto pp1, Posto p, pren_posto pp2
                where p.id= new.id and pp1!=pp2 and pp1.Posto=p.id
                and pp2.Posto=p.id);

Se isError = TRUE blocca l'operazione;
Altrimenti permetti l'operazione
```

7.6.5 V.Prenotazione.PostoInSala

Se prenoto per un Posto e per una rappresentazione allora quella rappresentazione deve trovarsi nella sala del posto

Trigger per il vincolo V.Prenotazione.PostoInSala:

Operazioni: inserimento o modifica in Prenotazione

Istante di invocazione: prima dell'operazione intercettata

Funzione:

Sia isError=FALSE;

Sia new l'ennupla che si sta inserendo oppure

l'ennupla risultato della modifica;

```
isError:= exists( select *
                  from Prenotazione p, Posto ps, Rappresentazione r,
                  Sala s, pren_posto pp
                  where p.id=new.id and pp.prenotazione = p.id
                  and pp.Posto= ps.id and p.Rappresentazione=r.id
                  and r.Sala=s.id and ps.Sala!=s.id);
```

Se isError = TRUE blocca l'operazione;

Altrimenti permetti l'operazione

7.7 Progettazione Funzionalità

Le Funzionalità da implementare nella base di dati sono:

- iscrizione
- prenotazione
- consulta_lista
- proponi_suggerimento
- calcola_posti_disponibili

7.7.1 calcola_posti_disponibili

`calcola_posti_disponibili(s: Intero): Intero`

1. Memorizza in Q il risultato della seguente query SQL:

```
SELECT
    (s.PostiTotali - prenotati.NumeroPrenotati) AS PostiDisponibili
FROM
    Sala s1
LEFT JOIN (
    SELECT
        p.Sala,
        COUNT(pp.Posto) AS NumeroPrenotati
    FROM
        Posto p
    JOIN
        prenotato pp ON p.id = pp.Posto
    GROUP BY
        p.Sala
) prenotati ON s1.id = prenotati.Sala
WHERE s.id = s1.id;
```

2. Se Q è vuoto generare l'errore 'La Sala non esiste'

3. Altrimenti restituisci il valore di Q

7.7.2 iscrizione

`iscrizione(n: Stringa, c: Stringa, cf: CF)`

1. Esegui il seguente comando SQL:

```
insert into Utente(Nome, Cognome, CF)
values(PAR_1, PAR_2, PAR_3)
dopo aver rimpiazzato PAR_1, PAR_2, PAR_3
con i valori dei parametri attuali rispettivamente n, c, cf
```

2. Se il comando precedente restituisce errore di vincolo di chiave violato restituisci l'errore 'Utente già esistente'

7.7.3 prenotazione

prenotazione(u: Utente, r: Rappresentazione, n: Intero)

1. Esegui il seguente comando SQL:

```
begin transiction
insert into Prenotazione(id, d, u, r)
values(sereal, now(), PAR_1, PAR_2)
dopo aver rimpiazzato PAR_1, PAR_2
con i valori dei parametri attuali rispettivamente u, r
salvando il valore di sereal in una variabile chiamata p
```

```
n volte
insert into pren_posto(p)
values(p)
end transticion
commit
```

2. Se il comando precedente restituisce errore di vincolo di chiave violato restituisci l'errore 'Pranotazione già esistente'

7.7.4 consulta_lista

consulta_lista(t: Stringa, g: Stringa, d: Data)

1. Memorizza in Q il risultato della seguente query SQL:

```
select *
from Rappresentazione r, Spettacolo s
where r.Spettacolo=s.id and s.Tipologia=t and s.Genere=g and r.DataOra=d
```

2. Se Q è vuoto generare l'errore:

'Nessuna Rappresentazione per i criteri inseriti'

3. Altrimenti restituisci il valore di Q

7.7.5 proponi_suggerimento

```
proponi_suggerimento(u: CF)
begin transiction
1. Memorizza in Q il risultato della seguente query SQL:
    select FIRST_ROW(s.Genere)
    from Spettacolo s, Utente u, Rappresentazione r, Prenotazione p, Genere g
    where u.cf = cf and
    p.Utente= u.cf and p.Rappresentazione=r.id and r.Spettacolo=s.id
    and s.genere= genere.nome
    order by p.DataOra desc
2. Memorizza in G il risultato della seguente query SQL:
    select *
    from Rappresentazione r, Spettacolo s,
    where s.genere=q and r.spettacolo= s.id and r.DataOra>now()+7

end transticion
commit
3. Se G è vuoto generare l'errore 'Nessun suggerimento disponibile'
4. Altrimenti ritorna G
```