SAPIENZA
UNIVERSITÀ DI ROMA

Basi di Dati, Modulo 2

Sapienza Università di Roma
Facoltà di Ing. dell'Informazione, Informatica e Statistica
Laurea in Informatica
Prof. Toni Mancini
http://tmancini.di.uniroma1.it

Progetto 20080918 (P.20080918)

Out!

Versione 2019-06-20

Indice

1 Introduzione

Si vuole progettare e realizzare *Out!*, un sistema Web che consenta agli utenti di consultare il programma di diversi cinema e teatri cittadini, e di prenotare gli spettacoli a cui desiderano assistere.

2 Specifica dei Requisiti

Out! deve consentire di mantenere informazioni su teatri e cinema e ogni altra sede di spettacoli associata al circuito. In particolare, di ogni sede interessa conoscere il nome e l'indirizzo. Le sedi possono essere provviste di più sale distinte, ognuna delle quali può ospitare uno spettacolo (ad es., cinema multisala, auditorium, etc.). Il sistema deve anche mantenere informazioni circa i posti a sedere nelle singole sale delle diverse sedi. Di ogni posto interessa conoscere i numeri di fila e colonna.

Out! deve inoltre rappresentare gli spettacoli in programmazione presso i diversi enti aderenti al circuito. Di ogni spettacolo interessa conoscere il titolo, la tipologia (concerto, rappresentazione teatrale, oppure film), il relativo genere e gli artisti che effettuano la performance, oltre che le date, gli orari e le sedi (con relativa sala) in cui viene rappresentato. Si osservi infatti che uno stesso spettacolo viene tipicamente rappresentato più volte; anche se tipicamente uno spettacolo viene rappresentato sempre nella stessa sede, il sistema deve permettere anche di rappresentare spettacoli itineranti. Per generalità dunque, si faccia il modo che il sistema possa mantenere informazioni circa la sede e la sala per ogni data in cui un certo spettacolo viene rappresentato.

Scopo principale del sistema è di consentire agli utenti di consultare il calendario degli spettacoli in programma, e di prenotare via web posti per gli spettacoli a cui desiderano assistere. In particolare, un utente deve poter usare il sistema per:

1. Iscriversi al servizio, fornendo i propri dati di interesse (nome, cognome, codice fiscale) che il sistema deve memorizzare;

2. Prenotare uno o più posti a prezzo pieno o ridotto di un dato settore per una data di uno spettacolo. I posti non sono liberi, pertanto il sistema deve assegnare, all'atto della prenotazione, un numero di posti sufficienti non ancora prenotati.[1]

[1] Si assuma per semplicità che i posti vengano assegnati in modo completamente arbitrario, e non dal "migliore" al "peggiore".

3. Consultare la lista degli spettacoli di una certa tipologia e genere (ad es., spettacoli teatrali/commedie) previsti in un certo giorno.

4. Ricevere dal sistema suggerimenti di nuovi spettacoli da vedere. In particolare, si richiede che il sistema segnali ad un utente l'insieme degli spettacoli programmati nei successivi 7 giorni, che sono dello stesso genere (anche se di tipologia diversa) dell'ultimo spettacolo da egli prenotato.

Parte A

Analisi Concettuale

A.1 Raffinamento dei Requisiti

A.1.1 Testo

Raffinare la specifica dei requisiti eliminando inconsistenze, omissioni o ridondanze e producendo un elenco numerato di requisiti il meno ambiguo possibile.

A.1.2 Soluzione

1. Di ogni sede (di spettacoli) sono di interesse:
   1.1. nome (una stringa)
   1.2. indirizzo (con via, numero civico, CAP, città, nazione)
   1.3. insieme delle sale, di ognuna delle sono di interesse:
      1.3.1. nome (una stringa)
      1.3.2. l'insieme dei settori, di ognuno dei quali sono di interesse:
         1.3.2.1. nome (una stringa)
         1.3.2.2. l'insieme dei posti a sedere, dei quali sono di interesse:
            1.3.2.2.1. numero di riga (intero > 0)
            1.3.2.2.2. numero di colonna (intero > 0)
2. Di ogni spettacolo sono di interesse:
   2.1. titolo
   2.2. tipologia (Req. 3.)
   2.3. genere (Req. 4.)
   2.4. artisti coinvolti
   2.5. le rappresentazioni (eventi, Req. 5.)
3. Di ogni tipologia di spettacolo è di interesse:
   3.1. nome (stringa)
4. Di ogni genere di spettacolo è di interesse:
   4.1. nome (stringa)
5. Di ogni evento (rappresentazione di uno spettacolo) sono di interesse:
   5.1. sala (Req. 1.3.) della sede (Req. 1.) dove è in programmazione
   5.2. data e ora di inizio
   5.3. il tariffario, ovvero un insieme di voci ognuna delle quali della forma:
      5.3.1. settore (Req. 1.3.2.) della sala dove l'evento è in programmazione
      5.3.2. tipologia di tariffa (Req. 8.)
      5.3.3. importo (in Euro).
      5.3.4. Un tariffario non può definire due voci che coincidono nella coppia settore/tipologia di tariffa.
6. Di ogni utente sono di interesse:

6.1. nome (stringa)

6.2. cognome (stringa)

6.3. codice fiscale (stringa secondo standard)

7. Di ogni prenotazione sono di interesse:

7.1. l'utente che l'ha effettuata (Req. 6.)

7.2. l'evento prenotato (Req. 5.)

7.3. l'insieme dei posti prenotati, di ognuno dei quali interessa:

7.3.1. i dettagli del posto a sedere (Req. 1.3.2.2.)

7.3.2. la tipologia di tariffa applicata (Req. 5.3.2.)

8. Di ogni tipologia di tariffa sono di interesse:

8.1. nome

9. Il sistema deve offrire le seguenti funzionalità:

9.1. Gli utenti web devono potersi iscrivere, fornendo i propri dati di interesse.

9.2. Gli utenti registrati devono poter effettuare prenotazioni per un evento, fornendo il numero richiesto di posti e, per ogni posto, la tipologia di tariffa richiesta. Il sistema deve assegnare un insieme non ancora prenotati di posti (in numero pari a quelli richiesti).

9.3. Gli utenti Web devono poter consultare la lista degli spettacoli di una data tipologia e genere previsti in un dato giorno.

9.4. Gli utenti registrati devono poter ricevere dal sistema suggerimenti di nuovi spettacoli da vedere. Il sistema segnalerà ad un utente l'insieme degli spettacoli programmati nei successivi 7 giorni che sono dello stesso genere dell'ultimo spettacolo da egli prenotato.

9.5. I membri dello staff di *Out!* devono poter:

9.5.1. Aggiungere e modificare spettacoli (Req. 2.) ed eventi in programmazione (Req. 5.)

9.5.2. Modificare l'insieme delle sedi (Req. 1.) del circuito, modificare i dati delle sedi e delle relative sale, settori, posti (Req. 1.)

9.5.3. Modificare l'insieme delle tipologie di tariffe (Req. 8.)

9.5.4. Modificare l'insieme dei generi di spettacolo (Req. 4.)

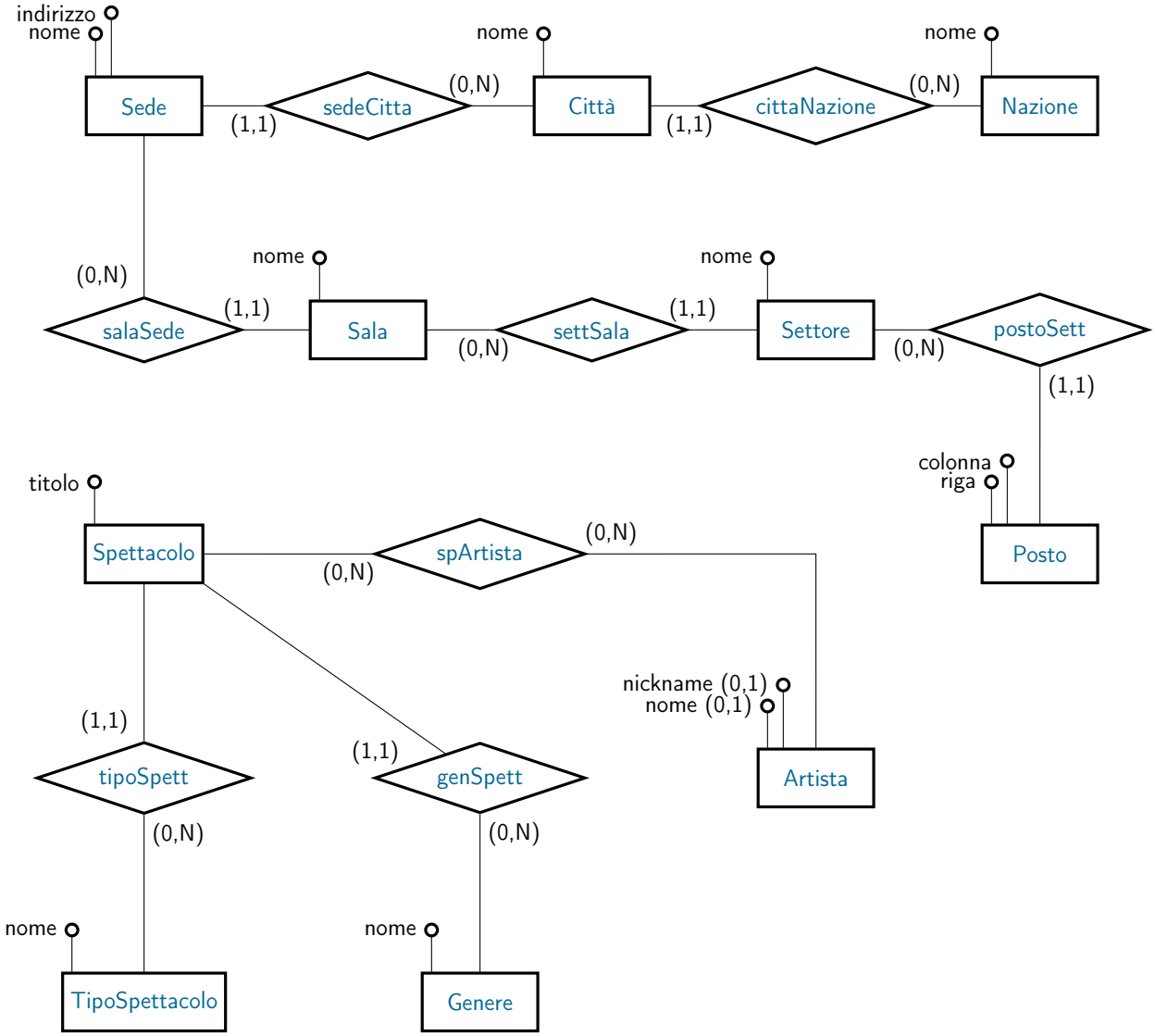
9.5.5. Modificare l'insieme delle tipologie di spettacolo (Req. 3.).

A.2 Diagramma ER

A.2.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti. In particolare, produrre il diagramma ER concettuale per l'applicazione e le specifiche dei dati per modellare Req. da 1. a 4.

A.2.2 Soluzione



Specifiche dei Dati

Entità Sede
Ogni istanza di questa entità rappresenta una sede (Req. 1.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della sede
indirizzo	Indirizzo		L'indirizzo della sede

Entità Città
Ogni istanza di questa entità rappresenta una città

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della città

Entità Nazione
Ogni istanza di questa entità rappresenta una nazione

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nazione

Entità Sala
Ogni istanza di questa entità rappresenta una sala di una Sede (Req. 1.3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della sala

Entità Settore
Ogni istanza di questa entità rappresenta un settore di una Sala (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del settore

Entità Posto
Ogni istanza di questa entità rappresenta un posto a sedere (Req. 1.3.2.2.).

attributo	dominio	molteplicità	descrizione
riga	intero > 0		Il numero di riga del posto
colonna	intero > 0		Il numero di colonna del posto

Entità Spettacolo
Ogni istanza di questa entità rappresenta uno Spettacolo (Req. 2.)

attributo	dominio	molteplicità	descrizione
titolo	stringa		Il titolo dello spettacolo

Entità TipoSpettacolo
Ogni istanza di questa entità rappresenta una tipologia di Spettacolo (Req. 2.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della tipologia di spettacoli

Entità Genere
Ogni istanza di questa entità rappresenta un genere di Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del genere

Entità Artista
Ogni istanza di questa entità rappresenta un artista (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	NomeCognome	(0,1)	Il nome dell'artista
nickname	stringa	(0,1)	Il nickname dell'artista

Vincoli:

[V.Artista.nome] Per ogni istanza dell'entità Artista deve essere definito o un valore per l'attributo nickname oppure un valore per l'attributo nome.

Relationship sedeCitta
Ogni istanza di questa relationship lega una Sede alla Città in cui si trova
Attributi: Nessuno

Relationship cittaNazione
Ogni istanza di questa relationship lega una Città alla Nazione in cui si trova
Attributi: Nessuno

Relationship salaSede
Ogni istanza di questa relationship lega una Sala alla sua Sede (Req. 1.3.)
Attributi: Nessuno

Relationship settSala
Ogni istanza di questa relationship lega un Settore alla sua Sala (Req. 1.3.2.)
Attributi: Nessuno

Relationship postoSett
Ogni istanza di questa relationship lega un Posto al suo Settore (Req. 1.3.2.2.)
Attributi: Nessuno

Relationship tipoSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo TipoSpettacolo (Req. 2.2.)
Attributi: Nessuno

Relationship genSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo Genere (Req. 2.3.)
Attributi: Nessuno

Relationship spArtista
Ogni istanza di questa relationship lega uno Spettacolo agli artisti che prendono parte ad esso (Req. 2.4.)
Attributi: Nessuno

Dominio NomeCognome
Il dominio è un record composto dai seguenti campi:

- nome: stringa
- cognome: stringa

Dominio Indirizzo
Il dominio è un record composto dai seguenti campi:

- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre

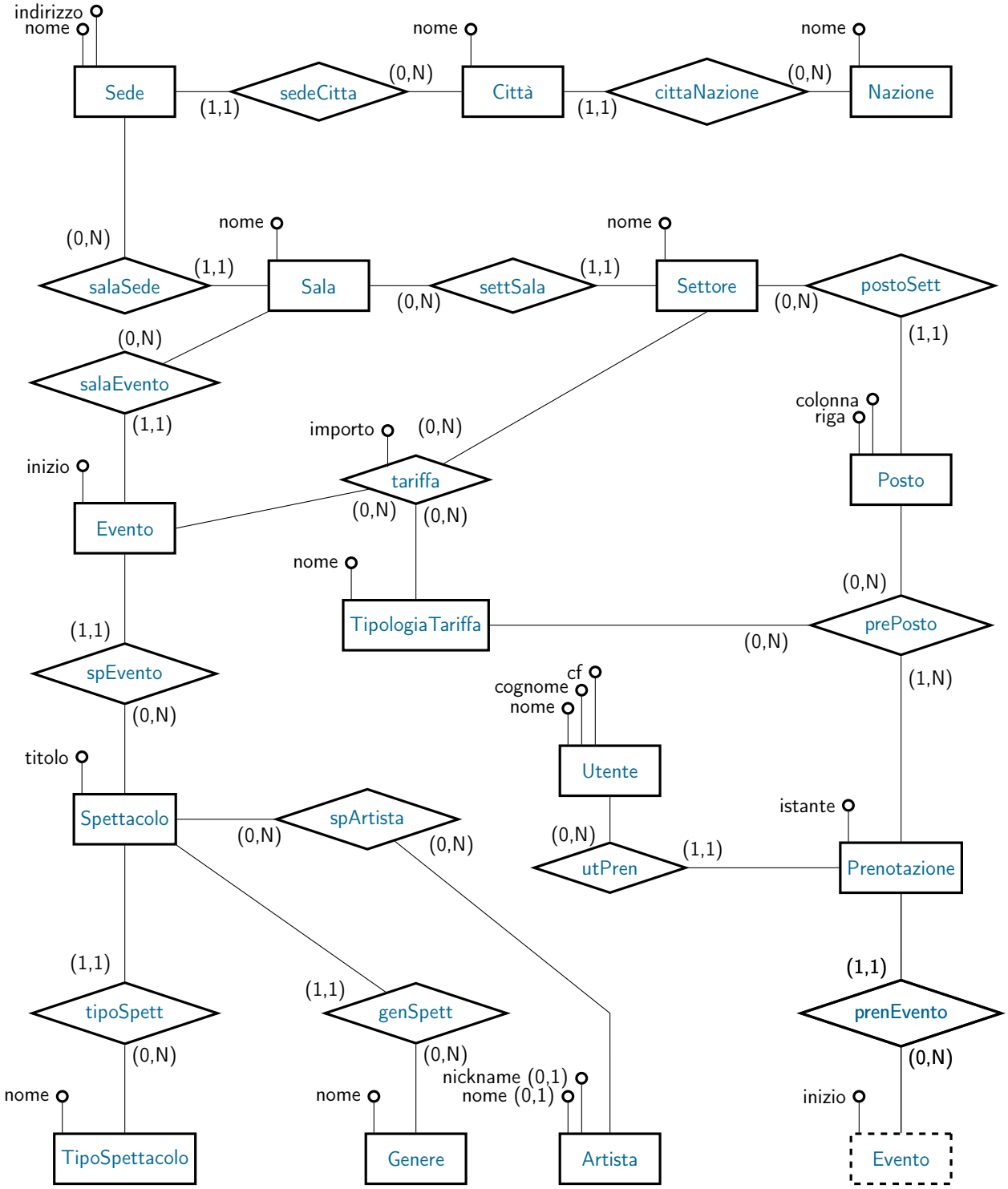
A.3 Estensione del diagramma ER

A.3.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti. In particolare, produrre il diagramma ER concettuale per l'applicazione e le specifiche dei dati per modellare Req. da 5. a 8.

Definire inoltre eventuali vincoli sui dati non esprimibili nel diagramma ER (vincoli esterni) in linguaggio naturale.

A.3.2 Soluzione



Specifiche dei Dati

Entità Sede
Ogni istanza di questa entità rappresenta una sede (Req. 1.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della sede
indirizzo	Indirizzo		L'indirizzo della sede

Entità Città
Ogni istanza di questa entità rappresenta una città

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della città

Entità Nazione
Ogni istanza di questa entità rappresenta una nazione

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nazione

Entità Sala
Ogni istanza di questa entità rappresenta una sala di una Sede (Req. 1.3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della sala

Entità Settore
Ogni istanza di questa entità rappresenta un settore di una Sala (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del settore

Entità Posto
Ogni istanza di questa entità rappresenta un posto a sedere (Req. 1.3.2.2.).

attributo	dominio	molteplicità	descrizione
riga	intero > 0		Il numero di riga del posto
colonna	intero > 0		Il numero di colonna del posto

Entità Spettacolo
Ogni istanza di questa entità rappresenta uno Spettacolo (Req. 2.)

attributo	dominio	molteplicità	descrizione
titolo	stringa		Il titolo dello spettacolo

Entità TipoSpettacolo
Ogni istanza di questa entità rappresenta una tipologia di Spettacolo (Req. 2.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della tipologia di spettacoli

Entità Genere
Ogni istanza di questa entità rappresenta un genere di Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del genere

Entità Artista
Ogni istanza di questa entità rappresenta un artista (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	NomeCognome	(0,1)	Il nome dell'artista
nickname	stringa	(0,1)	Il nickname dell'artista

Vincoli:

[V.Artista.nome] Per ogni istanza dell'entità Artista deve essere definito o un valore per l'attributo nickname oppure un valore per l'attributo nome.

Entità Evento
Ogni istanza di questa entità rappresenta un evento in cui viene rappresentato uno Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
inizio	dataora		l'orario di inizio dell'evento

Entità TipologiaTariffa
Ogni istanza di questa entità rappresenta una tipologia di tariffa relativa ad un Evento e ad un Settore (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della tipologia di tariffa

Entità Utente
Ogni istanza di questa entità rappresenta una tipologia di tariffa relativa ad un Evento e ad un Settore (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'utente
cognome	stringa		Il cognome dell'utente
cf	CodiceFiscale		Il codice fiscale dell'utente

Entità Prenotazione
Ogni istanza di questa entità rappresenta una prenotazione effettuata da un utente per un certo numero di posti per un evento

attributo	dominio	molteplicità	descrizione
istante	dataora		L'istante in cui l'utente ha effettuato la prenotazione

Vincoli:

[V.Prenotazione.PostiDistinti] Per ogni istanza pr : Prenotazione, non esistono due istanze di prenPosto nelle quali pr è coinvolta che coinvolgono la stessa istanza di Posto.

[V.Prenotazione.istante] Le prenotazioni devono essere effettuate prima dell'inizio del relativo evento. In dettaglio: per ogni istanza pr di entità Prenotazione,

siano:

- i_{pr} il valore dell'attributo istante di pr;
- i_e il valore dell'attributo inizio dell'istanza e di entità Evento associata a pr (ovvero tale per cui (pr, e) è istanza della relationship prenEvento).

Deve essere: $i_{pr} < i_e$.

Relationship sedeCitta
Ogni istanza di questa relationship lega una Sede alla Città in cui si trova
Attributi: Nessuno

Relationship cittaNazione
Ogni istanza di questa relationship lega una Città alla Nazione in cui si trova
Attributi: Nessuno

Relationship salaSede
Ogni istanza di questa relationship lega una Sala alla sua Sede (Req. 1.3.)
Attributi: Nessuno

Relationship settSala
Ogni istanza di questa relationship lega un Settore alla sua Sala (Req. 1.3.2.)
Attributi: Nessuno

Relationship postoSett
Ogni istanza di questa relationship lega un Posto al suo Settore (Req. 1.3.2.2.)
Attributi: Nessuno

Relationship tipoSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo TipoSpettacolo (Req. 2.2.)
Attributi: Nessuno

Relationship genSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo Genere (Req. 2.3.)
Attributi: Nessuno

Relationship spArtista
Ogni istanza di questa relationship lega uno Spettacolo agli artisti che prendono parte ad esso (Req. 2.4.)
Attributi: Nessuno

Relationship salaEvento
Ogni istanza di questa relationship lega un evento alla Sala in cui si tiene
Attributi: Nessuno

Relationship spEvento
Ogni istanza di questa relationship lega un evento allo Spettacolo di cui é una rappresentazione
Attributi: Nessuno

Relationship tariffa
Ogni istanza di questa relationship lega una tipologia di tariffa ad un evento e ad un settore, associando anche un importo di denaro

attributo	dominio	molteplicità	descrizione
importo	Denaro		L'importo di denaro associato alla tipologia di tariffa per il settore e l'evento

Vincoli:

[V.tariffa.Settore] Ogni istanza della relationship tariffa deve coinvolgere un evento e ad uno dei settori della sala dove si tiene e. In dettaglio: per ogni istanza (e : Evento, t : TipologiaTariffa, se : Settore) della relationship tariffa, sia sa : Sala tale che (e, sa) é istanza di salaEvento; si ha che: (sa, se) é istanza di settSala.

Relationship prePosto
Ogni istanza di questa relationship lega una prenotazione ai posti ai quali é relativa e la tipologia di tariffa scelta per l'acquisto
Attributi: Nessuno

Vincoli:

[V.prePosto.TariffaEsiste] Per ogni istanza (pr : Prenotazione, po : Posto, t : TipologiaTariffa) della relationship prePosto, deve esistere una tariffa per il settore di po, di tipologia t e per l'evento associato a pr (il che implica, dato [V.TipologiaTariffa.Settore], che po è in un settore della sala di e).

Relationship utPren
Ogni istanza di questa relationship lega un utente alle prenotazioni da lui/lei effettuate
Attributi: Nessuno

Relationship prenEvento
Ogni istanza di questa relationship lega una prenotazione all'evento a cui é relativa
Attributi: Nessuno

Dominio NomeCognome
Il dominio è un record composto dai seguenti campi:

- nome: stringa
- cognome: stringa

Dominio Indirizzo
Il dominio è un record composto dai seguenti campi:

- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre

Dominio Denaro
Il dominio è di tipo record composto dai seguenti campi:

- valuta: stringa di 3 caratteri che identifica una valuta secondo lo standard (ad es., "EUR", "USD")
- importo: reale ≥ 0.

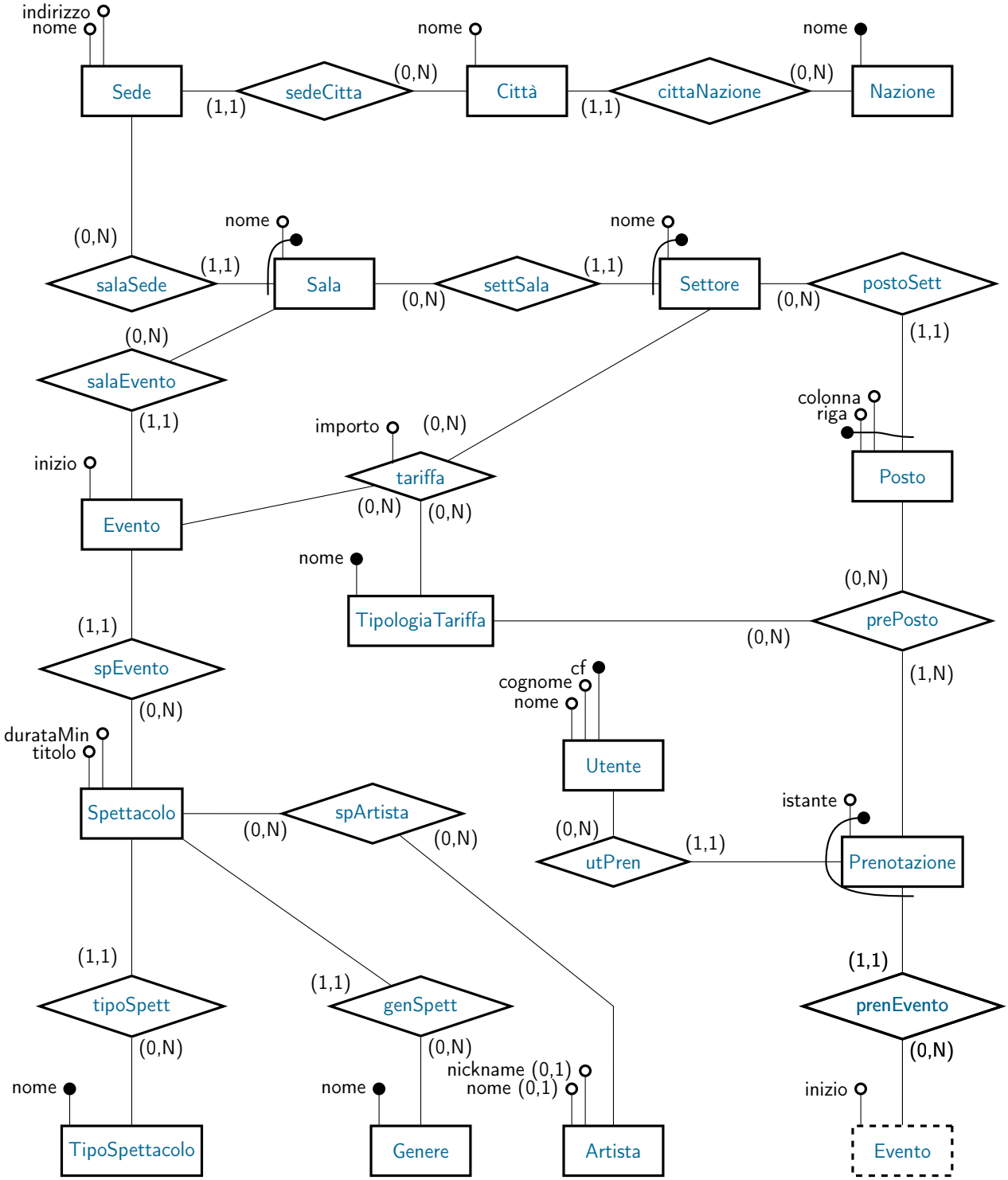
Dominio CodiceFiscale
Il dominio è composto dalle stringhe alfanumeriche di 16 caratteri che rispettano i vincoli dei codici fiscali italiani (http://it.wikipedia.org/wiki/Codice_fiscale#Generazione_del_codice_fiscale).

A.4 Vincoli esterni

A.4.1 Testo

Completare la fase di Analisi Concettuale dei dati. In particolare, finalizzare le specifiche dei dati definendo tutti i vincoli esterni in logica del primo ordine (anche raffinando il diagramma ER, se necessario).

A.4.2 Soluzione



Specifiche dei Dati

Entità Sede
Ogni istanza di questa entità rappresenta una sede (Req. 1.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della sede
indirizzo	Indirizzo		L'indirizzo della sede

Entità Città
Ogni istanza di questa entità rappresenta una città

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della città

Entità Nazione
Ogni istanza di questa entità rappresenta una nazione

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della nazione

Entità Sala
Ogni istanza di questa entità rappresenta una sala di una Sede (Req. 1.3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della sala

Entità Settore
Ogni istanza di questa entità rappresenta un settore di una Sala (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del settore

Entità Posto

Ogni istanza di questa entità rappresenta un posto a sedere (Req. 1.3.2.2.).

attributo	dominio	molteplicità	descrizione
riga	intero > 0		Il numero di riga del posto
colonna	intero > 0		Il numero di colonna del posto

Vincoli:

[V.Posto.NonDuePersonePerEvento] Per ogni istanza po : Posto, non esistono due istanze distinte (pr : Prenotazione, po : Posto, t : TipologiaTariffa) e (pr' : Prenotazione, po : Posto, t' : TipologiaTariffa) della relationship prePosto che coinvolgono il posto po insieme a prenotazioni pr e pr' relative allo stesso evento:

$$\forall \text{po } \text{posto}(\text{po}) \rightarrow \nexists \text{t}, \text{t'}, \text{pr}, \text{pr'}, \text{e} \quad \text{prePosto}(\text{pr}, \text{po}, \text{t}) \wedge \text{prePosto}(\text{pr'}, \text{po}, \text{t'}) \wedge \neg(\text{t} = \text{t'} \wedge \text{pr} = \text{pr'}) \wedge \text{prenEvento}(\text{pr}, \text{e}) \wedge \text{prenEvento}(\text{pr'}, \text{e}).$$

Entità Spettacolo

Ogni istanza di questa entità rappresenta uno Spettacolo (Req. 2.)

attributo	dominio	molteplicità	descrizione
titolo	stringa		Il titolo dello spettacolo
durataMin	intero > 0		La durata dello spettacolo (in minuti)

Entità TipoSpettacolo

Ogni istanza di questa entità rappresenta una tipologia di Spettacolo (Req. 2.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della tipologia di spettacoli

Entità Genere

Ogni istanza di questa entità rappresenta un genere di Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome del genere

Entità Artista

Ogni istanza di questa entità rappresenta un artista (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	NomeCognome	(0,1)	Il nome dell'artista
nickname	stringa	(0,1)	Il nickname dell'artista

Vincoli:

[V.Artista.nome] Per ogni istanza dell'entità Artista deve essere definito o un valore per l'attributo nickname oppure un valore per l'attributo nome:

$$\forall a\ \text{Artista}(a) \rightarrow \exists v\ \ (\text{nickname}(a, v) \vee \text{nome}(a, v))\,.$$

Entità Evento

Ogni istanza di questa entità rappresenta un evento in cui viene rappresentato uno Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
inizio	dataora		l'orario di inizio dell'evento

Entità TipologiaTariffa

Ogni istanza di questa entità rappresenta una tipologia di tariffa relativa ad un Evento e ad un Settore (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome della tipologia di tariffa

Entità Utente
Ogni istanza di questa entità rappresenta una tipologia di tariffa relativa ad un Evento e ad un Settore (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	stringa		Il nome dell'utente
cognome	stringa		Il cognome dell'utente
cf	CodiceFiscale		Il codice fiscale dell'utente

Entità Prenotazione
Ogni istanza di questa entità rappresenta una prenotazione effettuata da un utente per un certo numero di posti per un evento

attributo	dominio	molteplicità	descrizione
istante	dataora		L'istante in cui l'utente ha effettuato la prenotazione

Vincoli:

[V.Prenotazione.PostiDistinti] Per ogni istanza pr : Prenotazione, non esistono due istanze di prenPosto nelle quali pr è coinvolta che coinvolgono la stessa istanza di Posto.

Questo vincolo viene incorporato nel nuovo vincolo [V.Posto.NonDuePersonePerEvento], in quanto ne è un caso particolare.

[V.Prenotazione.istante] Le prenotazioni devono essere effettuate prima dell'inizio del relativo evento:

$$\forall pr, e, i_{pr}, i_e \quad \text{Prenotazione}(pr) \wedge \text{istante}(pr, i_{pr}) \wedge \text{Evento}(e) \wedge \text{prenEvento}(pr, e) \wedge \text{inizio}(e, i_e) \rightarrow i_{pr} < i_e.$$

Relationship sedeCitta
Ogni istanza di questa relationship lega una Sede alla Città in cui si trova
Attributi: Nessuno

Relationship cittaNazione
Ogni istanza di questa relationship lega una Città alla Nazione in cui si trova
Attributi: Nessuno

Relationship salaSede
Ogni istanza di questa relationship lega una Sala alla sua Sede (Req. 1.3.)
Attributi: Nessuno

Relationship settSala
Ogni istanza di questa relationship lega un Settore alla sua Sala (Req. 1.3.2.)
Attributi: Nessuno

Relationship postoSett
Ogni istanza di questa relationship lega un Posto al suo Settore (Req. 1.3.2.2.)
Attributi: Nessuno

Relationship tipoSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo TipoSpettacolo (Req. 2.2.)
Attributi: Nessuno

Relationship genSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo Genere (Req. 2.3.)
Attributi: Nessuno

Relationship spArtista
Ogni istanza di questa relationship lega uno Spettacolo agli artisti che prendono parte ad esso (Req. 2.4.)
Attributi: Nessuno

Relationship salaEvento
Ogni istanza di questa relationship lega un evento alla Sala in cui si tiene
Attributi: Nessuno

Vincoli:

[V.salaEvento.NonSovrapposti] Una sala non può ospitare eventi sovrapposti nel tempo.

$$\forall sa, e, e', s, s', i, i', d, d' \ \text{Sala}(sa) \wedge \text{Evento}(e) \wedge \text{Evento}(e') \wedge e \neq e' \wedge$$
$$\text{salaEvento}(sa, e) \wedge \text{salaEvento}(sa, e') \wedge$$
$$\text{spEvento}(s, e) \wedge \text{spEvento}(s', e') \wedge$$
$$\text{inizio}(e, i) \wedge \text{durataMin}(s, d) \wedge$$
$$\text{inizio}(e', i') \wedge \text{durataMin}(s', d') \rightarrow$$
$$\nexists t \ \text{dataora}(t) \wedge \ i \leq t \wedge t \leq i + d \wedge$$
$$i' \leq t \wedge t' \leq i' + d'.$$

Nella formula, il simbolo di funzione $+$, soggetto alla semantica reale, quando applicato ad una istanza i del dominio dataora e ad una durata d in minuti, restituisce l'istanza del dominio dataora che denota l'istante successivo ad i e distante da i esattamente d minuti.

Relationship spEvento
Ogni istanza di questa relationship lega un evento allo Spettacolo di cui é una rappresentazione
Attributi: Nessuno

Relationship tariffa
Ogni istanza di questa relationship lega una tipologia di tariffa ad un evento e ad un settore, associando anche un importo di denaro

attributo	dominio	molteplicità	descrizione
importo	Denaro		L'importo di denaro associato alla tipologia di tariffa per il settore e l'evento

Vincoli:

[V.tariffa.Settore] Ogni istanza della relationship tariffa deve coinvolgere un evento e ad uno dei settori della sala dove si tiene e:

$$\forall e, t, se, sa \ \text{tariffa}(e, t, se) \wedge \text{settSala}(se, sa) \rightarrow \text{salaEvento}(sa, e)$$

Relationship prePosto
Ogni istanza di questa relationship lega una prenotazione ai posti ai quali é relativa e la tipologia di tariffa scelta per l'acquisto
Attributi: Nessuno

Vincoli:

[V.prePosto.TariffaEsiste] Per ogni istanza (pr : Prenotazione, po : Posto, t : TipologiaTariffa) della relationship prePosto, deve esistere una tariffa per il settore di po, di tipologia t e per l'evento associato a pr (il che implica, dato [V.Tariffa.Settore], che po è in un Settore della sala di e):

$$\forall pr, po, t, se, e \quad Prenotazione(pr) \wedge Posto(po) \wedge TipologiaTariffa(t) \wedge Settore(se) \wedge Evento(e) \wedge prePosto(pr, po, t) \wedge postoSett(po, se) \wedge prenEvento(pr, e) \rightarrow tariffa(e, se, t).$$

Relationship utPren
Ogni istanza di questa relationship lega un utente alle prenotazioni da lui/lei effettuate
Attributi: Nessuno

Relationship prenEvento
Ogni istanza di questa relationship lega una prenotazione all'evento a cui é relativa
Attributi: Nessuno

Dominio NomeCognome
Il dominio è un record composto dai seguenti campi:

- nome: stringa
- cognome: stringa

Dominio Indirizzo
Il dominio è un record composto dai seguenti campi:

- via: stringa
- civico: intero > 0 (0,1)
- CAP: stringa di 5 cifre

Dominio Denaro
Il dominio è di tipo record composto dai seguenti campi:

- valuta: stringa di 3 caratteri che identifica una valuta secondo lo standard (ad es., "EUR", "USD")
- importo: reale ≥ 0.

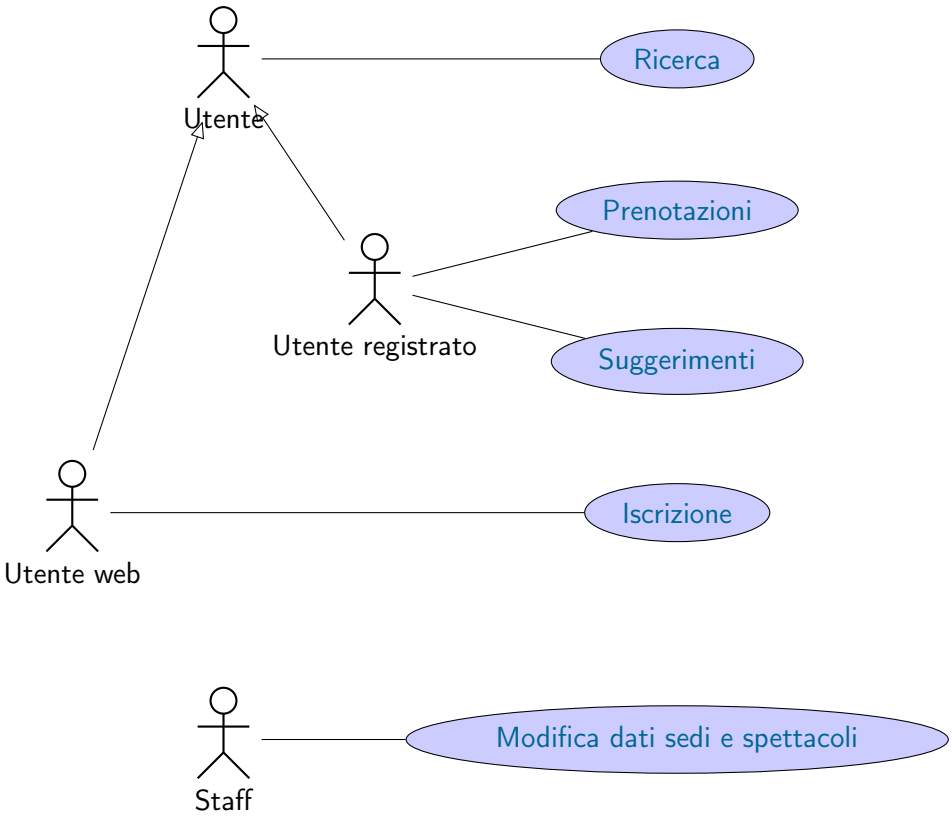
Dominio CodiceFiscale
Il dominio è composto dalle stringhe alfanumeriche di 16 caratteri che rispettano i vincoli dei codici fiscali italiani (http://it.wikipedia.org/wiki/Codice_fiscale#Generazione_del_codice_fiscale).

A.5 Diagamma UML degli Use-case

A.5.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti definendo il diagramma UML degli use-case.

A.5.2 Soluzione



A.6 Specifiche Concettuali degli Use-Case

A.6.1 Testo

Proseguire la fase di Analisi Concettuale dei requisiti definendo un diagramma UML degli use-case e formalizzando, in logica del primo ordine, le specifiche concettuali degli use-case.

A.6.2 Soluzione

Specifica Use-Case Ricerca

- **ricercaEventi(d: data, t: TipoSpettacolo, g: Genere) : Evento (0,N)** (Req. 9.3.)

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno:

$$\text{result} = \left\{ e \;\middle|\; \begin{array}{l} \text{Evento}(e) \;\wedge (\exists i\ \text{inizio}(e,i) \;\wedge \text{data}(i,d)) \;\wedge \\ \exists s\ \ \text{Spettacolo}(s)\ \text{spEvento}(s,e) \;\wedge \\ \qquad \text{tipoSpett}(t,s) \;\wedge \text{genSpett}(g,s) \end{array} \right\}$$

Specifica Use-Case Prenotazioni

- **prenota(u: Utente, e: Evento, posti : (p: Posto, t: TipologiaTariffa) (1,N)) : Prenotazione (Req. 9.2.)**

  precondizioni: I posti e le tipologie di tariffa richiesti sono disponibili per l'evento e. Formalmente, per ogni $(p, t) \in$ posti la seguente formula deve essere vera:

  $$\exists s \ \text{settore}(s) \ \wedge \text{postoSett}(p, s) \ \wedge \ \text{tariffa}(e, t, s) \ \wedge \ \text{disponibile}(p, e) = \text{true}$$

  postcondizioni:

  Modifica del livello estensionale dei dati: Il livello estensionale dei dati al termine dell'esecuzione della funzione differisce da quello di partenza come segue:

  Variazioni nel dominio di interpretazione: Un nuovo elemento α.

  Variazioni nelle ennuple di predicati:

  - Prenotazione(α).
  - istante(α, adesso).
  - Per ogni $(p, t) \in$ posti: prePosto(t, p, α).

  Valore di ritorno: result $= \alpha$.

- **disponibile(p: Posto, e: evento) : booleano**

  precondizioni: nessuna

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno: result = true se e solo se la seguente formula è vera:

  $$\nexists pr, t \ \text{Prenotazione}(pr) \ \wedge \text{prenEvento}(pr, e) \ \wedge \ \text{prePosto}(t, p, pr).$$

Specifica Use-Case Suggerimenti

- **suggerisci(u: Utente) : Evento (0,N) (Req. 9.4.)**

  precondizioni: L'utente u ha già effettuato delle prenotazioni:

  $$|\{p \mid \text{Prenotazione}(p) \wedge \text{utPren}(u, p)\}| \neq 0.$$

  postcondizioni:

  Modifica del livello estensionale dei dati: nessuna

  Valore di ritorno: Sia P l'insieme delle prenotazioni già effettuate da u con i loro relativi istanti:

  $$P = \{(p, i) \mid \text{Prenotazione}(p) \wedge \text{utPren}(u, p) \wedge \text{istante}(p, i)\}.$$

  Sia pu la prenotazione più recente in P. Formalmente, sia (pu, iu) una qualunque tra le coppie in P tale che:

  $$(pu, iu) \in \text{argmax}_i(P).$$

  Sia g il genere dello spettacolo associato alla prenotazione pu. Formalmente, g è tale da soddisfare la seguente formula:

  $$\exists e, s \quad \text{Evento}(e) \wedge \text{Spettacolo}(s) \wedge \text{prenEvento}(pu, e) \wedge \text{spEvento}(s, e) \wedge \text{genSpett}(g, s).$$

  $$\text{result} = \left\{ e \;\middle|\; \begin{array}{l} \text{Evento}(e) \wedge \\ \exists i\ \text{inizio}(e, i) \wedge \text{adesso} \leq i \wedge i \leq \text{adesso} + \text{“7 giorni”} \wedge \\ \exists s\ \text{spettEvento}(s, e) \wedge \text{genSpett}(g, s) \end{array} \right\}.$$

  Nella formula precedente, il simbolo di costante “7 giorni” è soggetto alla semantica di mondo reale.

Specifica Use-Case Iscrizione

- **iscrivi(cf: CodiceFiscale, n: stringa, c: stringa) : Utente (Req. 9.1.)**

  precondizioni: Non esiste alcuna istanza dell'entità Utente con il codice fiscale cf:

  $$\nexists \mathsf{u}\ \mathsf{Utente}(\mathsf{u}) \ \wedge \ \mathsf{cf}(\mathsf{u}, \mathsf{cf}).$$

  postcondizioni:

  Modifica del livello estensionale dei dati: Il livello estensionale dei dati al termine dell'esecuzione della funzione differisce da quello di partenza come segue:

  Variazioni nel dominio di interpretazione: Un nuovo elemento α.

  Variazioni nelle ennuple di predicati:

  - $\mathsf{Utente}(\alpha)$.
  - $\mathsf{cf}(\alpha, \mathsf{cf})$.
  - $\mathsf{nome}(\alpha, \mathsf{n})$.
  - $\mathsf{cognome}(\alpha, \mathsf{c})$.

  Valore di ritorno: $\mathsf{result} = \alpha$.

Specifica Use-Case Modifiche dati sedi e spettacoli (Req. 9.5.)

- La specifica di questo use-case è lasciata per esercizio.

Parte P

Progettazione della Base Dati e delle Funzionalità

P.1 Scelta del DBMS e Ristrutturazione del Diagramma ER e delle Specifiche dei Dati

P.1.1 Testo

Iniziare la fase di progettazione logica della base di dati decidendo il DBMS da utilizzare e ristrutturando il diagramma ER concettuale e le specifiche dei dati.

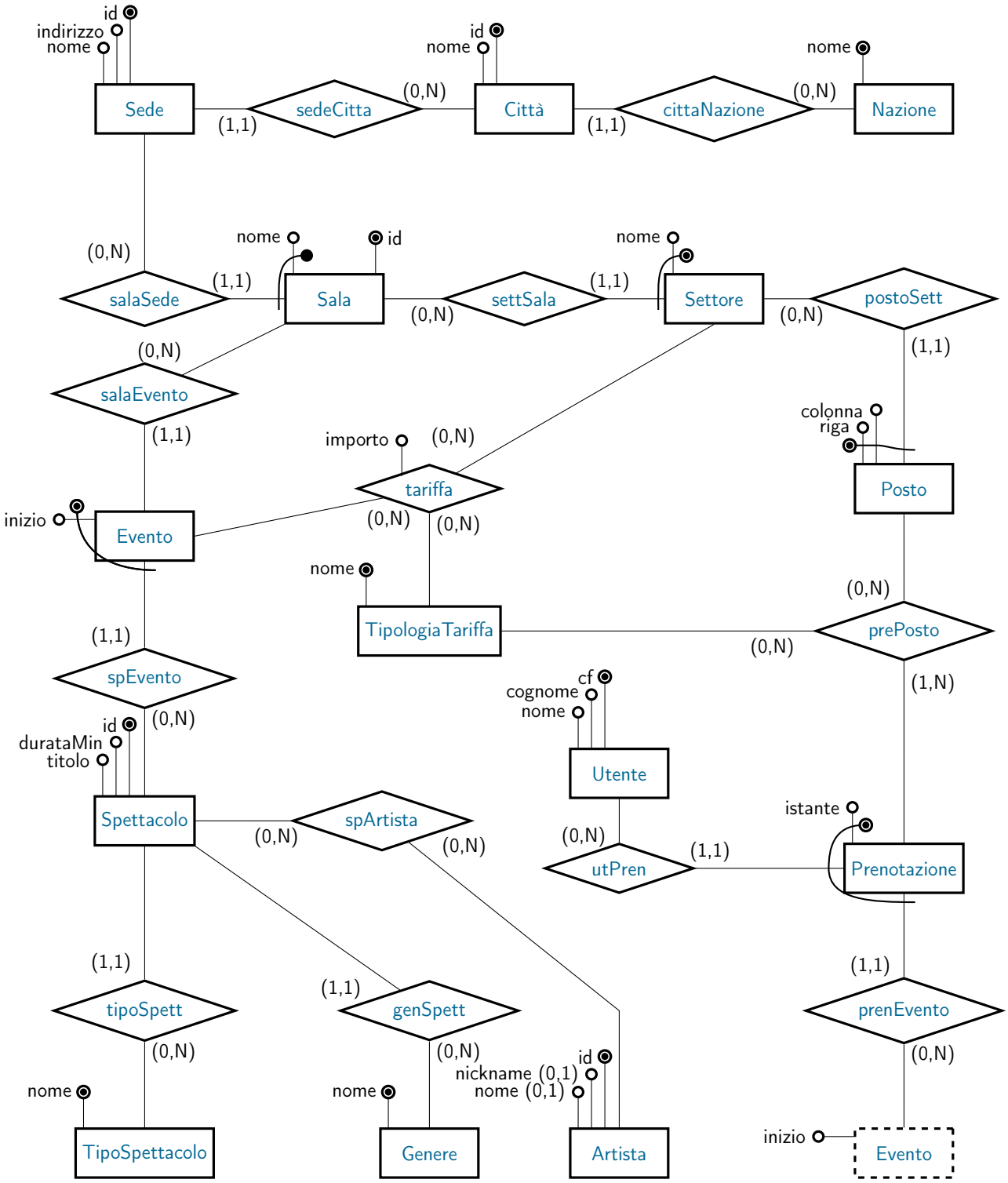
P.1.2 Soluzione

P.1.2.1 Scelta del DBMS

Si decide di utilizzare il DBMS PostgreSQL.

Nota: La scelta del DBMS è importante, in quanto può avere un impatto sul modo in cui vengono progettati i domini, i vincoli e le operazioni di use-case. In una fase di progetto completa, andrebbero prese anche altre decisioni, come l'architettura dell'applicazione ed il linguaggio di programmazione per quest'ultima. Tuttavia, dato lo scopo di questo corso, non prenderemo decisioni in tal senso.

P.1.2.2 Ristrutturazione del Diagramma ER e delle Specifiche dei Dati



Specifiche dei Dati

Entità Sede

Ogni istanza di questa entità rappresenta una sede (Req. 1.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della sede
indirizzo	Indirizzo		L'indirizzo della sede
id	integer		L'identificatore della sede

Entità Città

Ogni istanza di questa entità rappresenta una città

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della città
id	integer		L'identificatore della città

Entità Nazione

Ogni istanza di questa entità rappresenta una nazione

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della nazione

Entità Sala

Ogni istanza di questa entità rappresenta una sala di una Sede (Req. 1.3.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della sala
id	integer		L'identificatore della sala

Entità Settore

Ogni istanza di questa entità rappresenta un settore di una Sala (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome del settore

Entità Posto

Ogni istanza di questa entità rappresenta un posto a sedere (Req. 1.3.2.2.).

attributo	dominio	molteplicità	descrizione
riga	IntegerGZ		Il numero di riga del posto
colonna	IntegerGZ		Il numero di colonna del posto

Vincoli:

[V.Posto.NonDuePersonePerEvento] Per ogni istanza po : Posto, non esistono due istanze distinte (pr : Prenotazione, po : Posto, t : TipologiaTariffa) e (pr' : Prenotazione, po : Posto, t' : TipologiaTariffa) della relationship prePosto che coinvolgono il posto po insieme a prenotazioni pr e pr' relative allo stesso evento:

$$\forall po\ posto(po) \rightarrow \nexists t, t', pr, pr', e\ \ prePosto(pr, po, t) \wedge prePosto(pr', po, t') \wedge \neg(t = t' \wedge pr = pr') \wedge prenEvento(pr, e) \wedge prenEvento(pr', e).$$

Entità Spettacolo

Ogni istanza di questa entità rappresenta uno Spettacolo (Req. 2.)

attributo	dominio	molteplicità	descrizione
titolo	StringM		Il titolo dello spettacolo
durataMin	IntegerGZ		La durata dello spettacolo (in minuti)
id	integer		L'identificatore dello spettacolo

Entità TipoSpettacolo

Ogni istanza di questa entità rappresenta una tipologia di Spettacolo (Req. 2.2.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della tipologia di spettacoli

Entità Genere
Ogni istanza di questa entità rappresenta un genere di Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome del genere

Entità Artista
Ogni istanza di questa entità rappresenta un artista (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
nome	NomeCognome	(0,1)	Il nome dell'artista
nickname	StringM	(0,1)	Il nickname dell'artista
id	integer		L'identificatore dell'artista

Vincoli:

[V.Artista.nome] Per ogni istanza dell'entità Artista deve essere definito o un valore per l'attributo nickname oppure un valore per l'attributo nome:

$$\forall a\ \text{Artista}(a) \rightarrow \exists v\ \ (\text{nickname}(a, v) \vee \text{nome}(a, v))\ .$$

Entità Evento
Ogni istanza di questa entità rappresenta un evento in cui viene rappresentato uno Spettacolo (Req. 2.3.)

attributo	dominio	molteplicità	descrizione
inizio	timestamp		l'orario di inizio dell'evento

Entità TipologiaTariffa
Ogni istanza di questa entità rappresenta una tipologia di tariffa relativa ad un Evento e ad un Settore (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome della tipologia di tariffa

Entità Utente
Ogni istanza di questa entità rappresenta una tipologia di tariffa relativa ad un Evento e ad un Settore (Req. 1.3.2.)

attributo	dominio	molteplicità	descrizione
nome	StringM		Il nome dell'utente
cognome	StringM		Il cognome dell'utente
cf	CodiceFiscale		Il codice fiscale dell'utente

Entità Prenotazione
Ogni istanza di questa entità rappresenta una prenotazione effettuata da un utente per un certo numero di posti per un evento

attributo	dominio	molteplicità	descrizione
istante	timestamp		L'istante in cui l'utente ha effettuato la prenotazione

Vincoli:

[V.Prenotazione.istante] Le prenotazioni devono essere effettuate prima dell'inizio del relativo evento:

$$\forall pr, e, i_{pr}, i_e \quad \text{Prenotazione}(pr) \wedge \text{istante}(pr, i_{pr}) \wedge \text{Evento}(e) \wedge \text{inizio}(e, i_e) \rightarrow i_{pr} < i_e.$$

Relationship sedeCitta
Ogni istanza di questa relationship lega una Sede alla Città in cui si trova
Attributi: Nessuno

Relationship cittaNazione
Ogni istanza di questa relationship lega una Città alla Nazione in cui si trova
Attributi: Nessuno

Relationship salaSede
Ogni istanza di questa relationship lega una Sala alla sua Sede (Req. 1.3.)
Attributi: Nessuno

Relationship settSala
Ogni istanza di questa relationship lega un Settore alla sua Sala (Req. 1.3.2.)
Attributi: Nessuno

Relationship postoSett
Ogni istanza di questa relationship lega un Posto al suo Settore (Req. 1.3.2.2.)
Attributi: Nessuno

Relationship tipoSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo TipoSpettacolo (Req. 2.2.)
Attributi: Nessuno

Relationship genSpett
Ogni istanza di questa relationship lega uno Spettacolo al suo Genere (Req. 2.3.)
Attributi: Nessuno

Relationship spArtista
Ogni istanza di questa relationship lega uno Spettacolo agli artisti che prendono parte ad esso (Req. 2.4.)
Attributi: Nessuno

Relationship salaEvento
Ogni istanza di questa relationship lega un evento alla Sala in cui si tiene
Attributi: Nessuno

Vincoli:

[V.salaEvento.NonSovrapposti] Una sala non può ospitare eventi sovrapposti nel tempo:

$$\begin{aligned}\forall \text{sa, e,e', s, s', i, i', d, d' } & \text{Sala(sa)} \wedge \text{Evento(e)} \wedge \text{Evento(e')} \wedge \text{e} \neq \text{e'} \wedge \\ & \text{salaEvento(sa, e)} \wedge \text{salaEvento(sa, e')} \wedge \\ & \text{spEvento(s, e)} \wedge \text{spEvento(s', e')} \wedge \\ & \text{inizio(e, i)} \wedge \text{durataMin(s, d)} \wedge \\ & \text{inizio(e', i')} \wedge \text{durataMin(s', d')} \rightarrow \\ & \quad \nexists \text{t} \;\; \text{dataora(t)} \wedge \;\; \text{i} \leq \text{t} \wedge \text{t} \leq \text{i} + \text{d} \wedge \\ & \qquad\qquad\qquad\qquad \text{i'} \leq \text{t} \wedge \text{t'} \leq \text{i'} + \text{d'}.\end{aligned}$$

Nella formula, il simbolo di funzione $+$, soggetto alla semantica reale, quando applicato ad una istanza i del dominio dataora e ad una durata d in minuti, restituisce l'istanza del dominio dataora che denota l'istante successivo ad i e distante da i esattamente d minuti.

Relationship spEvento
Ogni istanza di questa relationship lega un evento allo Spettacolo di cui é una rappresentazione
Attributi: Nessuno

Relationship tariffa
Ogni istanza di questa relationship lega una tipologia di tariffa ad un evento e ad un settore, associando anche un importo di denaro

attributo	dominio	molteplicità	descrizione
importo	Denaro		L'importo di denaro associato alla tipologia di tariffa per il settore e l'evento

Relationship prePosto
Ogni istanza di questa relationship lega una prenotazione ai posti ai quali é relativa e la tipologia di tariffa scelta per l'acquisto
Attributi: Nessuno

Vincoli:

[V.prePosto.TariffaEsiste] Per ogni istanza (pr : Prenotazione, po : Posto, t : TipologiaTariffa) della relationship prePosto, deve esistere una tariffa per il settore di po, di tipologia t e per l'evento associato a pr (il che implica, dato [V.TipologiaTariffa.Settore], che po è in un settore della sala di e):

$$\forall pr, po, t, se, e$$
$$\text{Prenotazione}(pr) \wedge \text{Posto}(po) \wedge \text{Tipologia}(t) \wedge \text{Settore}(se) \wedge$$
$$\text{Evento}(e) \wedge \text{prePosto}(pr, po, t) \wedge$$
$$\text{postoSett}(po, se) \wedge \text{prenEvento}(pr, e) \rightarrow$$
$$\text{tariffa}(e, se, t).$$

Relationship utPren
Ogni istanza di questa relationship lega un utente alle prenotazioni da lui/lei effettuate
Attributi: Nessuno

Relationship prenEvento

Ogni istanza di questa relationship lega una prenotazione all'evento a cui é relativa
Attributi: Nessuno

Dominio StringM

Il dominio sarà definito mediante seguente comando SQL:

```
create domain StringM as varchar(200);
```

Dominio IntegerGZ

Il dominio sarà definito mediante seguente comando SQL:

```
create domain IntegerGZ as integer check (value > 0);
```

Dominio NomeCognome

Il dominio sarà definito mediante i seguenti comandi SQL:

```
create domain NomeCognome_Nome as StringM
    check (value is not null);

create domain NomeCognome_Cognome as StringM
    check (value is not null);

create type NomeCognome as
    (nome NomeCognome_Nome,
     cognome NomeCognome_Cognome);
```

Dominio Indirizzo

Il dominio sarà definito mediante i seguenti comandi SQL:

```
create domain IndirizzoVia as StringM
    check (value is not null);

create domain IndirizzoCivico as integer
    check (value > 0);

create domain IndirizzoCAP as char(5)
    check (value is not null and value ~ '^[0-9]*$');

create type Indirizzo as
    (via IndirizzoVia,
     civico IndirizzoCivico,
     cap IndirizzoCAP);
```

Dominio Denaro

Il dominio sarà definito mediante i seguenti comandi SQL:

```
create domain DenaroValuta as char(3)
    check (value is not null);

create domain DenaroImporto as real
    check (value is not null and value >= 0);

create type Denaro as
    (valuta DenaroValuta, importo DenaroImporto);
```

Dominio CodiceFiscale

Il dominio sarà definito mediante seguente comando SQL:

```
create domain CodiceFiscale as char(16)
    check (isCodiceFiscale(value));
```

dove isCodiceFiscale (**char**(16)): boolean è una opportuna funzione di DB (il cui progetto è lasciato per esercizio) che verifica che il parametro attuale soddisfi i vincoli dei codici fiscali italiani.

P.2 Schema Relazionale della Base Dati

P.2.1 Testo

Proseguire la fase di progettazione logica della base di dati producendo lo schema relazionale della base dati e i relativi vincoli (vincoli di chiave, di chiave primaria, di foreign key, di inclusione dovuti all'implementazione dei vincoli di molteplicità sulle relationship, di ennupla, di dominio) a partire dallo schema ER ristrutturato.

P.2.2 Soluzione

Disclaimer

Questa sezione ha bisogno di essere attentamente verificata.
Si pregano gli studenti di controllare la correttezza di tutti i dettagli e di riportare al docente eventuali errori rilevati. Grazie.

P.2.2.1 Definizione delle Relazioni Derivanti da Entità e Relationship Accorpate ad Entità

Sede (id:integer, nome:StringM, indirizzo:Indirizzo, citta:integer)
La relazione accorpa la relationship sedeCitta
[VincoloDB.1] *foreign key:* (citta) references Citta(id)
[VincoloDB.2] *serial:* I valori dell'attributo id sono generati automaticamente dal DBMS

Citta (id:integer, nome:StringM, nazione:StringM)
La relazione accorpa la relationship cittaNazione
[VincoloDB.3] *serial:* I valori dell'attributo id sono generati automaticamente dal DBMS
[VincoloDB.4] *foreign key:* (nazione) references Nazione(nome)

Nazione (nome:StringM)

Sala (id:integer, nome:StringM, sede:integer)
La relazione accorpa la relationship salaSede
[VincoloDB.5] *serial:* I valori dell'attributo id sono generati automaticamente dal DBMS
[VincoloDB.6] *chiave:* (nome, sede)
[VincoloDB.7] *foreign key:* (sede) references Sede(id)

Settore (sala:integer, nome:StringM)
La relazione accorpa la relationship settSala
[VincoloDB.8] *foreign key:* (sala) references Sala(id)

Posto (riga:IntegerGZ, colonna:IntegerGZ, sala:integer, settore:StringM)
La relazione accorpa la relationship postoSett
[VincoloDB.9] *foreign key:* (sala, settore) references Settore(sala, nome)

TipologiaTariffa (nome:StringM)

TipoSpettacolo (nome:StringM)

Genere (nome:StringM)

Spettacolo (id:integer, titolo:StringM, durataMin:IntegerGZ, tipo:StringM, genere:StringM)
La relazione accorpa le relationship tipoSpett e genSpett

[VincoloDB.10] *serial:* I valori dell'attributo id sono generati automaticamente dal DBMS
[VincoloDB.11] *foreign key:* (tipo) references TipoSpettacolo(nome)
[VincoloDB.12] *foreign key:* (genere) references Genere(nome)

Evento (spettacolo:integer, inizio:timestamp, sala:integer)
La relazione accorpa le relationship spEvento e salaEvento
[VincoloDB.13] *foreign key:* (spettacolo) references Spettacolo(id)
[VincoloDB.14] *foreign key:* (sala) references Sala(id)

Artista (id:integer, nome*:NomeCognome, nickname*:StringM)
[VincoloDB.15] *serial:* I valori dell'attributo id sono generati automaticamente dal DBMS
[VincoloDB.16] *ennupla:* [V.Artista.nome]: nome $\neq$ NULL $\vee$ nickname $\neq$ NULL

Utente (cf:CodiceFiscale, nome:StringM, cognome:StringM)

Prenotazione (utente:CodiceFiscale, istante:timestamp, eventoSpettacolo:integer, eventoInizio: timestamp)
La relazione accorpa le relationship utPren e prenEvento
[VincoloDB.17] *foreign key:* (utente) references Utente(cf)
[VincoloDB.18] *foreign key:* (eventoSpettacolo, eventoInizio) references Evento(spettacolo, inizio)
[VincoloDB.19] *inclusione:* (utente, istante, eventoSpettacolo, eventoInizio) $\subseteq$ prePosto(prenotazioneUtente, prenotazioneIstante, prenotazioneEventoSpettacolo, prenotazioneEventoInizio)
[VincoloDB.20] *ennupla:* [V.Prenotazione.istante]: istante $<$ eventoInizio

P.2.2.2 Definizione delle Relazioni Derivanti da Relationship non Accorpate ad Entità

tariffa (eventoSpettacolo:integer, eventoInizio:timestamp, tipologiaTariffa:StringM, settoreSala:integer, settoreNome:StringM, importo:Denaro)
[VincoloDB.21] *foreign key:* (eventoSpettacolo, eventoInizio) references Evento(spettacolo, inizio)
[VincoloDB.22] *foreign key:* (tipologiaTariffa) references TipologiaTariffa(nome)
[VincoloDB.23] *foreign key:* (settoreSala, settoreNome) references Settore(sala, nome)

prePosto (postoRiga:IntegerGZ, postoColonna:IntegerGZ, postoSala:integer, postoSettore:StringM, tipologiaTariffa:StringM, prenotazioneUtente:CodiceFiscale, prenotazioneIstante:timestamp, prenotazioneEventoSpettacolo:integer, prenotazioneEventoInizio:timestamp)
[VincoloDB.24] *foreign key:* (postoRiga, postoColonna, postoSala, postoSettore) references Posto(riga, colonna, sala, settore)
[VincoloDB.25] *foreign key:* (tipologiaTariffa) references TipologiaTariffa(nome)

[VincoloDB.26] *foreign key:* (prenotazioneUtente, prenotazioneIstante, prenotazioneEventoSpettacolo, prenotazioneEventoInizio) references Prenotazione(utente, istante, eventoSpettacolo, eventoInizio)

[VincoloDB.27] *[V.Posto.NonDuePersonePerEvento]:* Grazie ad accorti accorpamenti, questo vincolo viene implementato definendo una chiave su un sottoinsieme proprio degli attributi della chiave primaria di default (postoRiga, postoColonna, postoSala, postoSettore, tipologiaTariffa, prenotazioneUtente, prenotazioneIstante, prenotazioneEventoSpettacolo, prenotazioneEventoInizio). Tale nuova chiave diventa dunque la chiave primaria della relazione (mentre quella di default è semplicemente una super-chiave non $\subseteq$-minimale e dunque viene ignorata).

spArtista (spettacolo:integer, artista:integer)

[VincoloDB.28] *foreign key:* (spettacolo) references Spettacolo(id)

[VincoloDB.29] *foreign key:* (artista) references Artista(id)

P.3 Progettazione dei Vincoli Esterni

P.3.1 Testo

Proseguire la fase di progettazione logica della base di dati progettando come imporre i vincoli di integrità sui dati non esprimibili come vincoli di chiave, di chiave primaria, di foreign key, di ennupla, di dominio. Progettare inoltre come imporre i vincoli di inclusione definiti nello schema relazionale e non esprimibili come vincoli di foreign key.

P.3.2 Soluzione

Disclaimer

Questa sezione ha bisogno di essere attentamente verificata.
Si pregano gli studenti di controllare la correttezza di tutti i dettagli e di riportare al docente eventuali errori rilevati. Grazie.

Vincolo V.salaEvento.NonSovrapposti

Una sala non può ospitare eventi sovrapposti nel tempo.

Trigger

Operazioni: inserimento o modifica del valore dell'attributo inizio di una ennupla nella relazione Evento

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 new ← l'ennupla inserita o risultato della modifica;
/* isError *vale true se e solo se esiste un evento e diverso da new previsto nella stessa sala di new che si soprappone temporalmente a* new. */
2 isError ←

```
exists (
  select *
  from Spettacolo new_s, Evento e, Spettacolo s
  where new.spettacolo = new_s.id
    and e.sala = new.sala
    and e.spettacolo = s.id
    and (new.inizio, interval '1 minutes' * new_s.durataMin)
    overlaps (e.inizio, interval '1 minutes' * s.durataMin)
)
```

if isError **then** blocca l'operazione;
else permetti l'operazione;

Trigger

Operazioni: aumento del valore dell'attributo durataMin di una ennupla nella relazione Spettacolo

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 new ← l'ennupla inserita o risultato della modifica;
/* isError *vale true se e solo se esiste un evento new_e dello spettacolo new che si sovrappone temporalmente ad un altro evento other_new previsto nella stessa sala.* */
2 isError ←

```
exists (
  select *
  from Evento new_e, Evento other_e, Spettacolo other_s
  where new_e.spettacolo = new.id
    and new_e.sala = other_e.sala
    and other_e.spettacolo = other_s.id
    and (other_s.id <> new.id or other_e.inizio <> new_e.inizio)
    and (new_e.inizio, interval '1 minutes' * new.durataMin)
    overlaps (other_e.inizio, interval '1 minutes' * other_s.
        durataMin)
)
```

if isError **then** blocca l'operazione;
else permetti l'operazione;

Vincolo V.Tariffa.Settore

Ogni istanza della relationship tariffa deve coinvolgere un evento e ed uno dei settori della sala dove si tiene e.

Trigger

Operazioni: inserimento o modifica di una ennupla nella relazione tariffa

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 new ← l'ennupla da inserire o risultato della modifica;
/* isOK *vale true se e solo se l'evento associato alla tariffa* new *si tiene nella sala del settore associato alla tariffa* new. */
2 isOK ←

```
exists (
  select *
  from Evento e
  where e.spettacolo = new.eventoSpettacolo
   and e.inizio = new.eventoInizio
   and e.sala = new.settoreSala
)
```

if isOK **then** permetti l'operazione;
else blocca l'operazione;

Trigger

Operazioni: modifica del valore dell'attributo sala di una ennupla nella relazione Evento

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 new ← l'ennupla risultato della modifica;
/* isError *vale true se e solo se l'evento* new *ha delle tariffe associate.* */
2 isError ←

```
exists (
  select *
  from tariffa t
  where t.eventoSpettacolo = new.spettacolo
  and t.eventoInizio = new.inizio
)
```

if isError **then** blocca l'operazione;
else permetti l'operazione;

Vincolo [VincoloDB.19]

Vincolo di inclusione: Prenotazione(utente, istante, eventoSpettacolo, eventoInizio) $\subseteq$ prePosto(prenotazioneUtente, prenotazioneIstante, prenotazioneEventoSpettacolo, prenotazioneEventoInizio).

Trigger

Operazioni: inserimento o modifica dei valori di almeno uno degli attributi utente, istante, eventoSpettacolo, eventoInizio di una ennupla nella relazione Prenotazione

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 new ← l'ennupla inserita o risultato della modifica;
/* isOK *vale true se e solo se esiste una ennupla nella relazione prePosto che permette a new di soddisfare il vincolo di inclusione.* */
2 isOK ←

```
exists (
  select *
  from prePosto
  where prenotazioneUtente = new.utente
    and prenotazioneIstante = new.istante
    and prenotazioneEventoSpettacolo = new.eventoSpettacolo
    and prenotazioneEventoInizio = new.eventoInizio
)
```

if isOK **then** permetti l'operazione;
else blocca l'operazione;

Politica di accesso ai dati Impedire la modifica del valore degli attributi prenotazioneUtente, prenotazioneIstante, prenotazioneEventoSpettacolo, prenotazioneEventoInizio per le ennuple della relazione prePosto:

```
revoke update on prePosto from <utenti>;
grant update(postoRiga, postoColonna, postoSala,
   postoSettore, tipologiaTariffa) on prePosto to <
   utenti>;
```

dove <utenti> definisce la lista degli utenti delle applicazioni che dovranno interagire con la base dati.

Alternativamente (se il DBMS utilizzato non dovesse supportare politiche di accesso ai dati a livello di singoli attributi) è possibile intercettare, mediante un altro trigger, i tentativi di **update** su questi attributi e generare una eccezione.

Trigger

Operazioni: cancellazione di una ennupla nella relazione prePosto

Istante di Invocazione: dopo l'operazione intercettata

Funzione:

1 old ← l'ennupla cancellata;
/* isOK *vale true se e solo se esistono ancora ennuple nella relazione prePosto associate alla stessa prenotazione associata ad old.* */
2 isOK ←

```
exists (
  select *
  from prePosto
  where prenotazioneUtente = old.prenotazioneUtente
    and prenotazioneIstante = old.prenotazioneIstante
    and prenotazioneEventoSpettacolo = old.
        prenotazioneEventoSpettacolo
    and prenotazioneEventoInizio = old.prenotazioneEventoInizio
)
```

if isOK **then** permetti l'operazione;
else blocca l'operazione;

Vincolo V.prePosto.TariffaEsiste

Vincolo di inclusione: prePosto(tipologiaTariffa, postoSala, postoSettore, prenotazioneEventoSpettacolo, prenotazioneEventoInizio) $\subseteq$ tariffa(tipologiaTariffa, settoreSala, settoreNome, eventoSpettacolo, eventoInizio).

Trigger

Operazioni: inserimento o modifica dei valori di almeno uno degli attributi tipologiaTariffa, postoSala, postoSettore di una ennupla nella relazione prePosto

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 new ← l'ennupla inserita o risultato della modifica;
/* isOK *vale true se e solo se esiste una ennupla nella relazione tariffa che permette a new di soddisfare il vincolo di inclusione.* */
2 isOK ←

```
exists (
  select *
  from tariffa
  where tipologiaTariffa = new.tipologiaTariffa
    and settoreSala = new.postoSala
    and settoreNome = new.postoSettore
    and eventoSpettacolo = new.prenotazioneEventoSpettacolo
    and eventoInizio = new.prenotazioneEventoInizio
)
```

if isOK **then** permetti l'operazione;
else blocca l'operazione;

Politica di accesso ai dati Impedire le modifiche delle ennuple della relazione tariffa:

```
revoke update on tariffa from <utenti>;
```

dove <utenti> definisce la lista degli utenti delle applicazioni che dovranno interagire con la base dati.

Politica di accesso ai dati Impedire la modifica del valore degli attributi prenotazioneUtente, prenotazioneIstante, prenotazioneEventoSpettacolo, prenotazioneEventoInizio per le ennuple della relazione prePosto:

```
revoke update on prePosto from <utenti>;
grant update(postoRiga, postoColonna, postoSala,
    postoSettore, tipologiaTariffa) on prePosto to <
    utenti>;
```

dove <utenti> definisce la lista degli utenti delle applicazioni che dovranno interagire con la base dati.

Alternativamente (se il DBMS utilizzato non dovesse supportare politiche di accesso ai dati a livello di singoli attributi) è possibile intercettare, mediante un altro trigger, i tentativi di **update** su questi attributi e generare una eccezione.

Trigger

Operazioni: cancellazione di una ennupla nella relazione tariffa

Istante di Invocazione: prima dell'operazione intercettata

Funzione:

1 old ← l'ennupla da cancellare;

/* isError *vale true se e solo se esistono ennuple nella relazione prePosto associate alla tariffa old.* */

2 isError ←

```
exists (
  select *
  from prePosto
  where tipologiaTariffa = old.tipologiaTariffa
    and postoSala = old.settoreSala
    and postoSettore = old.settoreNome
    and eventoSpettacolo = old.prenotazioneEventoSpettacolo
    and eventoInizio = old.prenotazioneEventoInizio
)
```

if isError **then** blocca l'operazione;
else permetti l'operazione;