

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

PROGETTO D'ESAME DI BASI DI DATI

PROGETTAZIONE ED IMPLEMENTAZIONE DI UNA  
BASE DI DATI RELAZIONALE PER LA GESTIONE  
DI CONFERENZE SCIENTIFICHE

**Relatore**

Professoressa Mara SANGIOVANNI

**Candidati**

Antonio CAPORASO

matr: N86003458

Giorgio DI FUSCO

matr: N86004389

Anno Accademico 2022-2023

# Indice

<b>1</b>	<b>Traccia</b>	<b>6</b>
1.1	Output attesi dal committente . . . . .	6
<b>2</b>	<b>Progettazione</b>	<b>7</b>
2.1	Analisi dei dati . . . . .	7
2.2	Schema concettuale . . . . .	7
2.3	Ristrutturazione dello schema concettuale . . . . .	7
2.3.1	Rimozione degli attributi multivalore . . . . .	7
2.3.2	Rimozione classi di associazione . . . . .	7
2.3.3	Rimozione generalizzazioni . . . . .	7
2.3.4	Scelta degli identificatori principali . . . . .	8
2.4	Progettazione logica . . . . .	9
2.4.1	Traduzione delle classi . . . . .	9
2.4.2	Traduzione delle associazioni . . . . .	10
2.4.3	Schema logico . . . . .	11
<b>3</b>	<b>Implementazione fisica</b>	<b>15</b>
3.1	Definizione delle tabelle . . . . .	15
3.2	Definizione dei trigger . . . . .	20
3.2.1	Check_Programma_Entry . . . . .	20
3.2.2	Check_Data_Intervento, Check_Data_Intervallo, Check_Data_Evento	22
3.2.3	Create_Programma_Sessione . . . . .	23
3.2.4	Check_Sala_Sessione . . . . .	24
3.2.5	Check_Data_Sessione . . . . .	24
3.2.6	Check_Coordinatore_Sessione . . . . .	25
3.2.7	Create_Comitati_Conferenza . . . . .	26
3.2.8	Check_Comitati_Conferenza . . . . .	26
3.2.9	Check_Sala_Sessione_Unica . . . . .	27
3.2.10	Check_Organizzatore_Comitato . . . . .	28
3.2.11	Delete_Sessioni_Conferenza . . . . .	29
3.3	Funzioni e procedure . . . . .	30
3.3.1	Show_Conferenze_By_Date(date,date) . . . . .	30
3.3.2	Show_Conferenze_By_Sede(integer) . . . . .	30
3.3.3	Show_comitato_scientifico(integer) . . . . .	30
3.3.4	Show_comitato_locale(integer) . . . . .	31
3.3.5	Show_Partecipanti(integer) . . . . .	31
3.3.6	Show_Sessioni(integer) . . . . .	31
3.3.7	Show_interventi_sessione(integer) . . . . .	32
3.3.8	Show_intervalli_sessione(integer) . . . . .	32

3.3.9	Show_eventi_sociali_sessione(integer)	33
3.3.10	Show_keynote_sessione(integer)	34
3.3.11	Show_Programma(integer)	34
3.3.12	Add_Intervento(text,text,text,int,interval)	35
3.3.13	Add_Intervallo(text,int,interval)	36
3.3.14	Add_evento(text,int,interval)	37
3.3.15	Add_Conferenza_Details(text,timestamp,timestamp,integer,text)	38
3.3.16	Add_ente(integer, integer)	38
3.3.17	Add_Sponsorizzazione(integer,numeric,char(3),integer)	38
3.3.18	Add_Sessioni(text,timestamp,timestamp, integer,integer)	39
3.3.19	Add_Partecipante(integer, integer)	39
3.3.20	Add_Enti(integer,text)	39
3.3.21	Add_Conferenza(text,timestamp,timestamp,integer, text, text)	40
3.3.22	Slitta_Conferenza(interval)	40
3.3.23	Show_members()	43
3.3.24	Show_percentage_interventi(int,int)	43
3.3.25	Show_percentage(int)	44
3.4	Definizione delle viste	44
3.4.1	SediView	44
3.4.2	Conferenze_Sede	45
3.4.3	Interventi_Speaker	45
3.4.4	Partecipanti_Sessioni	45
3.4.5	Partecipanti_Conferenze	45
<b>A</b>	<b>Dizionari</b>	<b>46</b>
A.1	Dizionario dei dati	46
A.2	Dizionario delle associazioni	49
A.3	Dizionario dei vincoli	51

# Elenco delle figure

2.1	Schema concettuale del problema . . . . .	12
2.2	Ristrutturazione dello schema concettuale . . . . .	13
2.3	Schema logico . . . . .	14

# Elenco delle tabelle

2.1	Entità del problema . . . . .	8
-----	-------------------------------	---

# Elenco dei listati

Scripts/Tables.sql . . . . .	15
3.1 check_programma_entry . . . . .	20
3.2 check_data_intervento . . . . .	22
3.3 create_programma_sessione . . . . .	23
3.4 check_data_sessione . . . . .	24
3.5 check_coordinatore_sessione . . . . .	25
3.6 create_comitati_conferenza . . . . .	26
3.7 check_comitati_conferenza . . . . .	26
3.8 Check_sala_sessione_unica . . . . .	27
3.9 Check_organizzatori_comitato . . . . .	28
3.10 delete_sessioni_conferenza . . . . .	29

# Capitolo 1

## Traccia

Si sviluppi un sistema informativo, composto da una base di dati relazionale e da un applicativo Java dotato di GUI (Swing o JavaFX), per la gestione di **conferenze scientifiche**.

Ogni conferenza ha una data di inizio e di fine, una collocazione (sede, indirizzo), uno o più enti che la organizzano, degli sponsor (che coprono in parte le spese), una descrizione, ed un gruppo di organizzatori, che può essere distinto in comitato scientifico e comitato locale (che si occupa cioè della logistica). Di ognuno degli organizzatori, così come di tutti i partecipanti, si riportano titolo, nome, cognome, email ed istituzione di afferenza.

Ogni conferenza può avere una o più sessioni, anche in parallelo fra loro. Ogni sessione ha una locazione all'interno della sede. Per ogni sessione c'è un programma, che prevede la presenza di un coordinatore (chair) che gestisce la sessione, ed eventualmente di un keynote speaker (un partecipante di particolare rilievo invitato dagli organizzatori). Ogni sessione avrà quindi una successione di interventi ad orari predefiniti e di specifici partecipanti. Per ogni intervento si conserva un abstract (un breve testo in cui viene spiegato il contenuto del lavoro presentato).

Si deve poter considerare la presenza di spazi di intervallo (coffee breaks, pranzo) ma anche la presenza di eventi sociali (cene, gite, etc).

### 1.1 Output attesi dal committente

1. Documento di Design della base di dati:
  - (a) Class Diagram della base di dati.
  - (b) Dizionario delle Classi, delle Associazioni e dei Vincoli.
  - (c) Schema Logico con descrizione di Trigger e Procedure individuate.
2. File SQL contenenti:
  - (a) Creazione della struttura della base di dati.
  - (b) Popolamento del DB.
  - (c) (Facoltativo, ma apprezzato) README contenente i commenti all'SQL.

## Capitolo 2

# Progettazione

### 2.1 Analisi dei dati

Le entità che possono essere individuate nel problema sono elencate all'interno della Tabella 2.1.

### 2.2 Schema concettuale

Nella Figura 2.1 è presente lo schema concettuale della base di dati descritta nella sezione 1.

### 2.3 Ristrutturazione dello schema concettuale

#### 2.3.1 Rimozione degli attributi multivalore

All'interno del diagramma delle classi mostrato in Figura 2.1 sono presenti vari attributi multivalore. Per ciascuno di essi sono state fatte le seguenti valutazioni:

1. Si partiziona l'attributo *Indirizzo* presente in SEDE suddividendolo in vari campi *Via*, *Civico*, *Cap*, *City*, *Provincia* e *Nazione* e creando una nuova entità chiamata INDIRIZZO.
2. Si è deciso di partizionare l'attributo *Valuta* presente nella classe di associazione SPONSORIZZAZIONE creando una nuova classe chiamata VALUTA.

#### 2.3.2 Rimozione classi di associazione

All'interno dello schema concettuale è presente la classe di associazione SPONSORIZZAZIONE all'interno dell'associazione [\*...\*] tra CONFERENZA e SPONSOR. Nello schema ristrutturato questa è stata rimossa reificandola e scindendo l'associazione in due associazioni di tipo [1..\*].

#### 2.3.3 Rimozione generalizzazioni

Per quanto riguarda la rimozione delle generalizzazioni presenti nello schema concettuale:

1. Nel caso delle entità COMITATO SCIENTIFICO e COMITATO LOCALE che specializzano la classe COMITATO si è optato per l'accorpamento delle classi figlie all'interno della super-classe attraverso la specifica di una enumerazione chiamata COMITATO\_ST composta dai campi *Scientifico* e *Locale*;
2. Nel caso delle entità PRANZO e COFFEE BREAK che specializzano la classe INTERVALLO si è adottato la stessa politica.



Entità	Descrizione
<b>Conferenza</b>	Per le conferenze delle quali si vuole poter gestire le informazioni. Di ogni conferenza si conservano il <i>nome</i> , l' <i>inizio</i> e la <i>fine</i> e una <i>descrizione</i> .
<b>Ente</b>	Per gli enti che organizzano le conferenze scientifiche. Di ogni ente si conserva il <i>nome</i> e la <i>sigla</i> .
<b>Sponsor</b>	Per gli sponsor che coprono le spese della conferenza. Di ogni sponsor si conserva il <i>nome</i> .
<b>Comitato</b>	Per i gruppi di organizzatori che si occupano della gestione della conferenza. Si distinguono in comitati <i>scientifici</i> e <i>locali</i> .
<b>Organizzatore</b>	Per i membri dei comitati. Di ogni organizzatore si riportano <i>titolo</i> , <i>nome</i> , <i>cognome</i> , <i>email</i> ed <i>istituzione di afferenza</i> .
<b>Sede</b>	Per descrivere il luogo dove si tengono le varie conferenze. Di ogni sede si conservano il <i>nome</i> , l' <i>indirizzo</i> e la <i>città</i> .
<b>Sala</b>	Per tenere traccia dell'ubicazione delle varie sessioni. Di ogni sala si conserva il <i>nome della sala</i> e la sua <i>capacità</i> .
<b>Sessione</b>	Per rappresentare le sessioni di una conferenza. Per ogni sessione si riporta il <i>titolo</i> , un <i>coordinatore</i> , data e orario d' <i>inizio</i> e di <i>fine</i> .
<b>Programma</b>	Per il programma di ciascuna sessione. Ogni programma specifica la presenza di un <i>keynote speaker</i> , ovvero un partecipante di rilievo.
<b>Intervento</b>	Per i vari interventi di una sessione. Per ogni intervento si conserva un <i>abstract</i> , il partecipante ( <i>speaker</i> ) che effettua l'intervento e l' <i>orario</i> dello stesso.
<b>Partecipante</b>	Per i partecipanti delle varie sessioni. Ogni partecipante ha gli stessi attributi degli organizzatori.
<b>Intervallo</b>	Per descrivere i vari intervalli presenti all'interno di una sessione. Questi possono essere di due tipologie: <i>coffee break</i> oppure dei <i>pranzi</i> . Per ogni intervallo si riporta l' <i>orario</i> .
<b>Evento sociale</b>	Per i vari eventi sociali previsti all'interno di una sessione. Questi possono essere di varia natura. Come per gli intervalli se ne riporta l' <i>orario</i> .

**Tabella 2.1:** *Entità del problema*

### 2.3.4 Scelta degli identificatori principali

Risulta conveniente ai fini di una migliore traduzione delle associazioni l'introduzione di chiavi surrogate per ogni entità. Tali chiavi altro non saranno che identificativi numerici interi del tipo *Id\_NomeEntità*, eccezion fatta per l'entità VALUTA la quale viene identificata univocamente da una stringa di tre caratteri stando allo standard ISO 4217<sup>1</sup>.

<sup>1</sup>ISO 4217 è uno standard internazionale che descrive codici di tre lettere per definire i nomi delle valute, stabilito dall'Organizzazione internazionale per la normazione (ISO), che viene usato comunemente nel sistema bancario e nel mondo economico, nonché nella stampa specializzata.

## 2.4 Progettazione logica

Una volta aver ristrutturato lo schema concettuale mostrato in Figura 2.1 si procede traducendo le varie associazioni descritte in Figura 2.2. Iniziamo col tradurre direttamente tutte le classi. Man mano che si andranno a tradurre le varie associazioni andremo a modificare la struttura dei vari schemi relazionali laddove necessario.

### 2.4.1 Traduzione delle classi

Si ha quindi:

Indirizzo

<u>Id_Indirizzo</u>	Via	Civico	CAP	City	Provincia	Nazione
---------------------	-----	--------	-----	------	-----------	---------

ENTE

<u>id_ente</u>	nome	sigla
----------------	------	-------

SEDE

<u>id_sede</u>	nome
----------------	------

SPONSOR

<u>id_Sponsor</u>	Nome
-------------------	------

COMITATO

<u>id_Comitato</u>	Tipologia
--------------------	-----------

ORGANIZZATORE

<u>id_Organizzatore</u>	Nome	Cognome	Titolo	Email
-------------------------	------	---------	--------	-------

SALA

<u>id_Sala</u>	Nome	Capienza
----------------	------	----------

CONFERENZA

<u>id_Conferenza</u>	Nome	Descrizione	Inizio	Fine
----------------------	------	-------------	--------	------

PARTECIPANTE

<u>id_Partecipante</u>	Nome	Cognome	Titolo	Email
------------------------	------	---------	--------	-------

SESSIONE

<u>id_Sessione</u>	Nome	Inizio	Fine
--------------------	------	--------	------

VALUTA

<u>Iso</u>	Nome	Simbolo
------------	------	---------

SPEAKER

<u>id_Speaker</u>	Nome	Cognome	Titolo	Email
-------------------	------	---------	--------	-------

PROGRAMMA

<u>Id_Programma</u>
---------------------

INTERVALLO

<u>id_Intervallo</u>	Tipologia	Inizio	Fine
----------------------	-----------	--------	------

INTERVENTO

<u>id_Intervento</u>	Titolo	Abstract	Inizio	Fine
----------------------	--------	----------	--------	------

EVENTO

<u>id_Evento</u>	Tipologia	Inizio	Fine
------------------	-----------	--------	------

## 2.4.2 Traduzione delle associazioni

### Traduzione delle associazioni molti a molti

Traduciamo le associazioni \*.\* mediante la realizzazioni di apposite tabelle ponte. Si ha allora:

1. L'associazione ENTECONFERENZA tra ENTE e CONFERENZA:

ENTECONFERENZA

<u>id_ente</u>	<u>id_conferenza</u>
----------------	----------------------

2. L'associazione ORGANIZZATORECOMITATO tra ORGANIZZATORE e COMITATO:

ORGANIZZATORECOMITATO

<u>id_organizzatore</u>	<u>id_comitato</u>
-------------------------	--------------------

3. L'associazione PARTECIPANTESESSIONE tra PARTECIPANTE e SESSIONE:

PARTECIPANTESESSIONE

<u>id_Partecipante</u>	<u>id_Sessione</u>
------------------------	--------------------

### Traduzione delle associazioni uno a molti

Per ciascuna delle associazioni binarie di tipo uno a molti si identificano le entità deboli e quelle forti che partecipano all'associazione. Per tradurre l'associazione in relazioni basterà includere la chiave surrogata dell'entità forte all'interno della relazione dell'entità debole. Avremo quindi:

1. Associazioni di composizione:

- (a) Una sede è composta da più sale quindi:

SALA

<u>id_Sala</u>	Nome	Capienza	<u>id_sede</u>
----------------	------	----------	----------------

- (b) Una conferenza è composta da più sessioni:

SESSIONE

<u>id_Sessione</u>	Nome	Inizio	Fine	<u>id_conferenza</u>
--------------------	------	--------	------	----------------------

- (c) Un programma è composto da interventi, intervalli ed eventi:

INTERVALLO

<u>id_Intervallo</u>	Tipologia	Inizio	Fine	<u>id_programma</u>
----------------------	-----------	--------	------	---------------------

INTERVENTO

<u>id_Intervento</u>	Titolo	Abstract	Inizio	Fine	<u>id_programma</u>
----------------------	--------	----------	--------	------	---------------------

EVENTO

<u>id_Evento</u>	Tipologia	Inizio	Fine	<u>id_programma</u>
------------------	-----------	--------	------	---------------------

- Un partecipante, uno speaker ed un organizzatore appartengono ad una istituzione, ovvero un ENTE:

SPEAKER

<u>id_Speaker</u>	Nome	Cognome	Titolo	Email	<u>id_ente</u>
-------------------	------	---------	--------	-------	----------------

PARTECIPANTE

<u>id_Speaker</u>	Nome	Cognome	Titolo	Email	<u>id_ente</u>
-------------------	------	---------	--------	-------	----------------

ORGANIZZATORE

<u>id_Speaker</u>	Nome	Cognome	Titolo	Email	<u>id_ente</u>
-------------------	------	---------	--------	-------	----------------

- Ogni intervento ha uno speaker che lo effettua:

INTERVENTO

<u>id_Intervento</u>	<u>id_speaker</u>	Titolo	Abstract	Inizio	Fine	<u>id_programma</u>
----------------------	-------------------	--------	----------	--------	------	---------------------

- Una sala può ospitare più sessioni:

SESSIONE

<u>id_Sessione</u>	Nome	Inizio	Fine	<u>id_sala</u>	<u>id_conferenza</u>
--------------------	------	--------	------	----------------	----------------------

- Una sede può ospitare più conferenze:

CONFERENZA

<u>id_Conferenza</u>	Nome	Descrizione	Inizio	Fine	<u>id_sede</u>
----------------------	------	-------------	--------	------	----------------

- Una conferenza ha due comitati, uno scientifico ed uno locale:

CONFERENZA

<u>id_Conferenza</u>	Nome	Descrizione	Inizio	Fine	<u>id_sede</u>	<u>id_com_locale</u>	<u>id_com_scientifico</u>
----------------------	------	-------------	--------	------	----------------	----------------------	---------------------------

## Traduzione delle associazioni uno a uno

Si ha:

- Ogni sessione ha un coordinatore:

SESSIONE

<u>id_Sessione</u>	Nome	Inizio	Fine	<u>id_sala</u>	<u>id_conferenza</u>	<u>id_coordinatore</u>
--------------------	------	--------	------	----------------	----------------------	------------------------

- Ogni programma si riferisce ad una sessione e ad un keynote speaker:

PROGRAMMA

<u>Id_Programma</u>	<u>id_Sessione</u>	<u>id_keynote</u>
---------------------	--------------------	-------------------

- Ogni sede ha un indirizzo:

SEDE

<u>id_sede</u>	nome	indirizzo
----------------	------	-----------

### 2.4.3 Schema logico

Nella Figura 2.3 è raffigurato lo schema logico risultante.

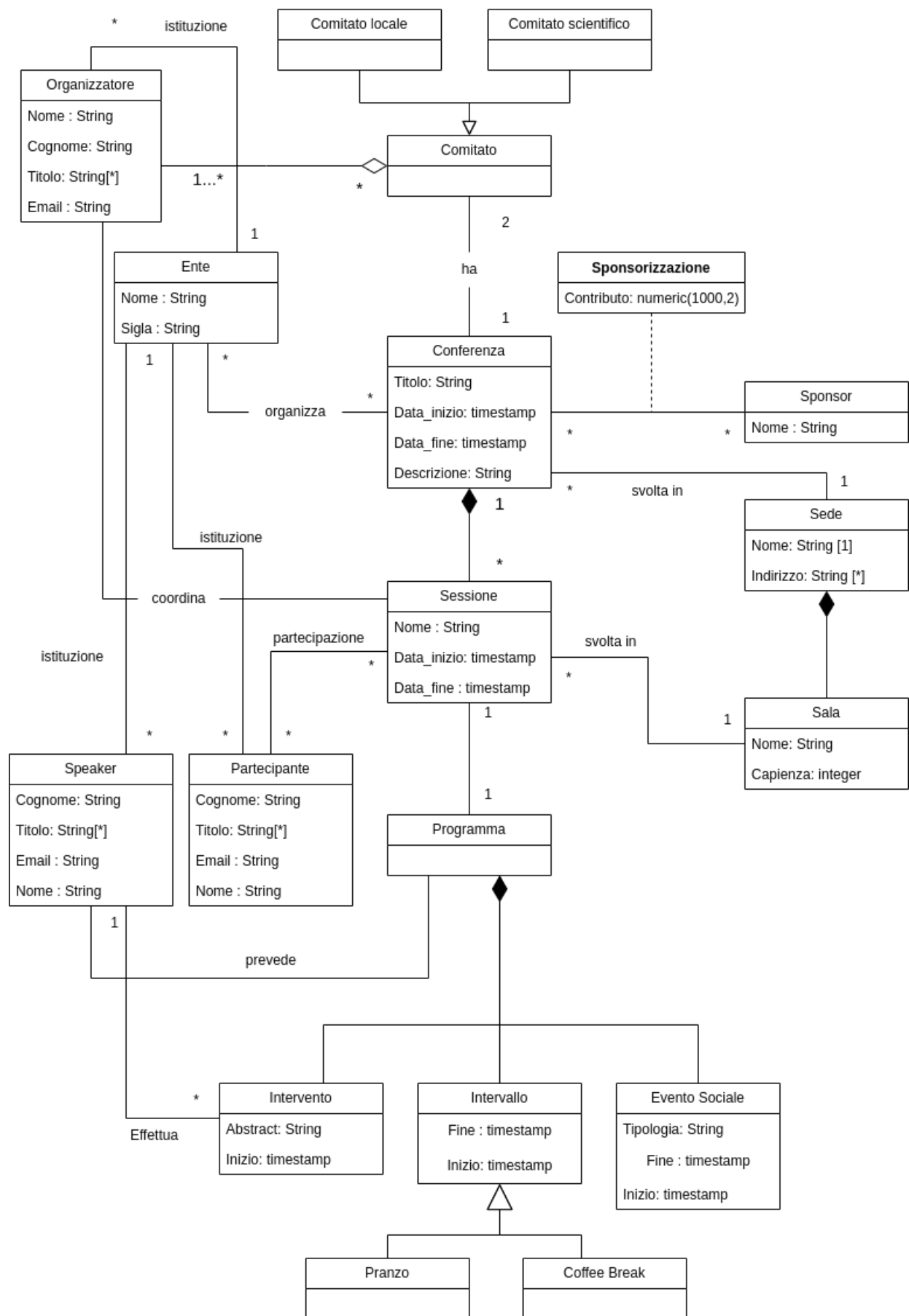
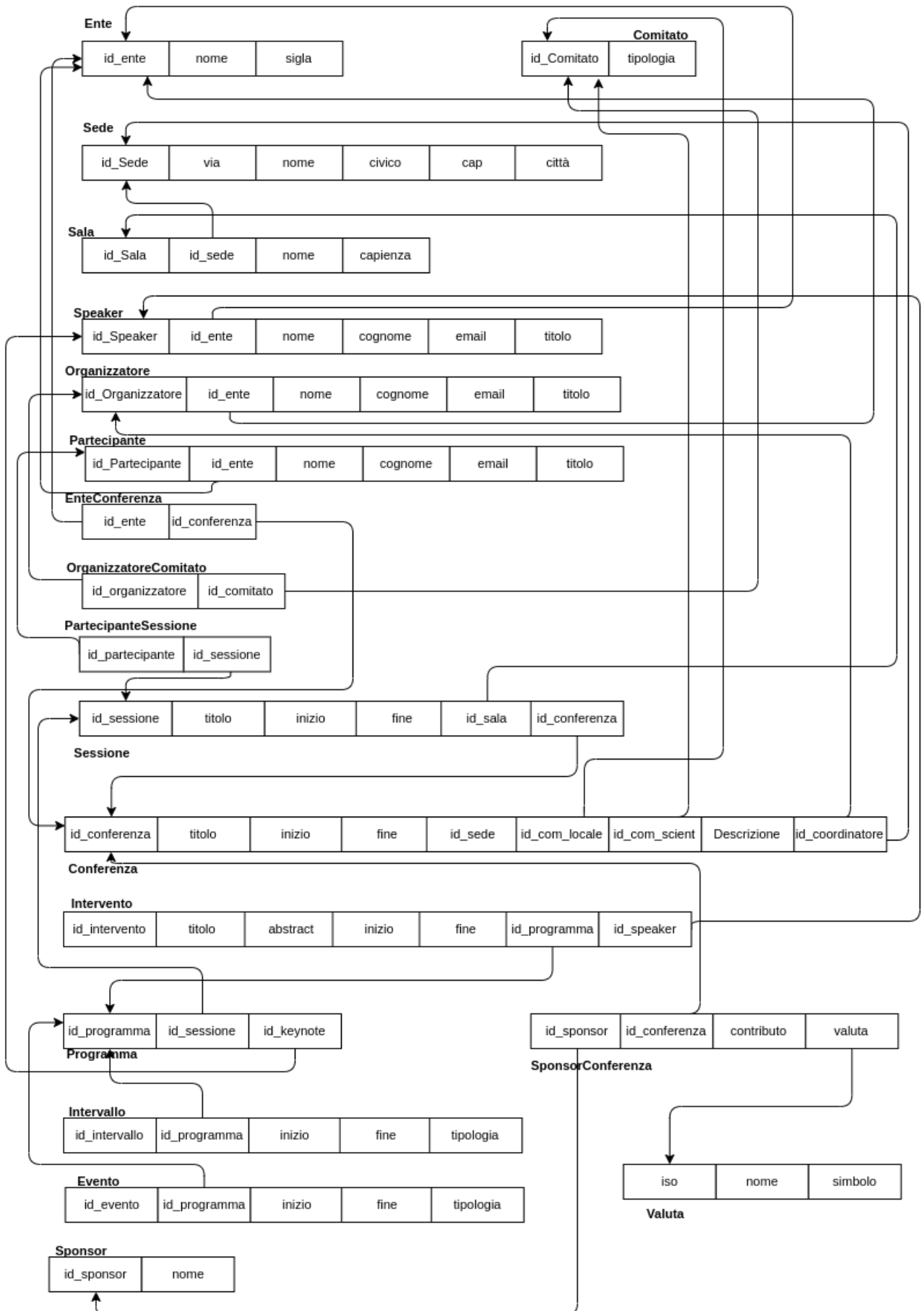


Figura 2.1: Schema concettuale del problema



Figura 2.3: Schema logico



## Capitolo 3

# Implementazione fisica

### 3.1 Definizione delle tabelle

```
1 create schema conference;
2 set search_path to conference;
3
4 create sequence ente_id_ente_seq;
5 create or replace function nextval_ente() returns text as $$
6 begin
7     return 'IST'||to_char(nextval('ente_id_ente_seq'),'FM0000');
8 end;
9 $$
10 language plpgsql;
11
12 create table ente(
13 id_ente text primary key default nextval_ente(),
14 nome text not null unique,
15 sigla varchar(7) not null,
16 unique (nome,sigla)
17 );
18
19 create table indirizzo(
20     id_indirizzo serial primary key,
21     via text not null,
22     civico varchar(5) not null,
23     cap varchar(5) ,
24     city text not null,
25     provincia varchar(2) not null,
26     nazione text
27 );
28
29 create table sede(
30     id_sede serial primary key,
31     nome text ,
32     id_indirizzo integer references indirizzo(id_indirizzo) on
33     delete set null
34 );
```



```

34
35
36 create table sponsor(
37     id_sponsor serial primary key,
38     nome text not null
39 );
40
41
42 create type comitato_st as enum ('locale','scientifico');
43 create table comitato(
44     id_comitato serial primary key,
45     tipologia comitato_st not null
46 );
47
48
49 create type titolo_st as enum ('Dottore','Dottoressa','Professore',
50     ,'Professoressa','Assistente','Ricercatore','Ricercatrice','
51     Ingegnere');
52 create table organizzatore(
53     id_organizzatore serial primary key,
54     nome text not null,
55     cognome text not null,
56     titolo titolo_st,
57     email text not null unique,
58     id_ente text references ente(id_ente) on delete cascade
59 );
60
61
62 create table sala(
63     id_sala serial primary key,
64     nome text not null,
65     capienza integer not null,
66     id_sede integer references sede(id_sede) on delete cascade
67 );
68
69
70 create table conferenza(
71     id_conferenza serial primary key,
72     titolo text not null,
73     descrizione text not null,
74     inizio timestamp not null,
75     fine timestamp not null,
76     id_sede integer references sede(id_sede) on delete set null,
77     comitato_s integer references comitato(id_comitato) on delete
78     set null,
79     comitato_l integer references comitato(id_comitato) on delete
80     set null,
81     -- Vincolo di integrita': la conferenza deve iniziare prima di
82     finire

```

```

78     check (inizio <= fine),
79     -- Vincolo di integrita': la conferenza deve iniziare in
    futuro
80     check (inizio >= now())
81 );
82
83
84 create table partecipante(
85     id_partecipante serial primary key,
86     nome text not null,
87     cognome text not null,
88     titolo titolo_st,
89     email text not null unique,
90     id_ente text references ente(id_ente) on delete set null
91 );
92
93
94 create table sessione(
95     id_sessione serial primary key,
96     titolo text not null,
97     inizio timestamp not null,
98     fine timestamp not null,
99     id_coordinatore integer references organizzatore(
100 id_organizzatore) on delete set null,
101     id_conferenza integer references conferenza(id_conferenza) on
102 delete cascade,
103     id_sala integer references sala(id_sala) on delete set null,
104     -- Vincolo di integrita': la sessione deve iniziare prima di
    finire
105     check (inizio <= fine)
106 );
107
108 create table partecipazione(
109     id_partecipante integer references partecipante(
110 id_partecipante) on delete cascade,
111     id_sessione integer references sessione(id_sessione) on delete
112 cascade,
113     -- Vincolo di unicity: un partecipante puo' partecipare ad
    una sessione una sola volta
114     unique (id_partecipante,id_sessione)
115 );
116
117 create table ente_conferenza(
118     id_ente text references ente(id_ente) on delete cascade,
119     id_conferenza integer references conferenza(id_conferenza) on
120 delete cascade,
121     -- Vincolo di unicity: un ente puo' organizzare una
    conferenza una sola volta
122     unique (id_ente,id_conferenza)

```

```

118 );
119
120 create table valuta(
121     iso char(3) primary key,
122     nome text not null,
123     simbolo text not null
124 );
125
126 create table sponsor_conferenza(
127     id_sponsor integer references sponsor(id_sponsor) on delete
128     cascade not null,
129     contributo numeric(1000,2) not null,
130     valuta char(3) references valuta(iso) not null,
131     id_conferenza integer references conferenza(id_conferenza) on
132     delete cascade not null,
133     -- Vincolo di unicit : uno sponsor puo' sponsorizzare una
134     conferenza una sola volta
135     unique (id_sponsor,id_conferenza)
136 );
137
138 create sequence speaker_id_speaker_seq;
139 create or replace function nextval_speaker() returns text as $$
140 begin
141     return 'SPK' || to_char(nextval('speaker_id_speaker_seq'),'
142     FM0000');
143 end;
144 $$
145 language plpgsql;
146
147 create table speaker(
148     id_speaker text primary key default nextval_speaker(),
149     nome text not null,
150     cognome text not null,
151     titolo titolo_st,
152     email text not null unique,
153     id_ente text references ente(id_ente) on delete set null
154 );
155
156 create sequence programma_id_programma_seq;
157 create or replace function nextval_programma() returns text as $$
158 begin
159     return 'PRG' || to_char(nextval('programma_id_programma_seq'),'
160     FM0000');
161 end;
162 $$
163 language plpgsql;
164
165 create table programma(
166     id_programma text primary key default nextval_programma(),

```

```

162     id_sessione integer references sessione(id_sessione) on delete
        cascade not null,
163     id_keynote text references speaker(id_speaker) on delete set
        null,
164     unique (id_programma, id_sessione)
165 );
166
167 create sequence intervento_id_intervento_seq;
168 create or replace function nextval_intervento() returns text as
        $$
169 begin
170     return 'INT' || to_char(nextval('intervento_id_intervento_seq'),
        'FM0000');
171 end;
172 $$
173 language plpgsql;
174
175 create table intervento(
176     id_intervento text primary key default nextval_intervento(),
177     titolo text not null,
178     abstract text not null,
179     inizio timestamp not null,
180     fine timestamp not null,
181     id_speaker text references speaker(id_speaker) on delete
        cascade,
182     id_programma text references programma(id_programma) on delete
        cascade not null,
183     -- Vincolo di unicita': uno speaker puo' intervenire in una
        sessione una sola volta
184     unique (id_speaker, id_programma),
185     -- Vincolo di integrita': l'intervento deve iniziare prima di
        finire
186     check (inizio <= fine)
187 );
188
189 create type intervallo_st as enum ('pranzo', 'coffee break');
190 create sequence intervallo_id_intervallo_seq;
191 create or replace function nextval_intervallo() returns text as
        $$
192 begin
193     return 'BRK' || to_char(nextval('intervallo_id_intervallo_seq'),
        'FM0000');
194 end;
195 $$
196 language plpgsql;
197
198 create table intervallo(
199     id_intervallo text primary key default nextval_intervallo(),
200     tipologia intervallo_st not null,

```

```

201     inizio timestamp not null,
202     fine timestamp not null,
203     -- Vincolo di integrita': l'intervallo deve iniziare prima di
finire
204     check (inizio <= fine),
205     id_programma text references programma(id_programma) on delete
cascade not null
206 );
207
208 create sequence evento_id_evento_seq;
209 create or replace function nextval_evento() returns text as $$
210 begin
211     return 'EVT' || to_char(nextval('evento_id_evento_seq'), 'FM0000'
);
212 end;
213 $$
214 language plpgsql;
215
216 create table evento(
217     id_evento text primary key default nextval_evento(),
218     tipologia text not null,
219     inizio timestamp not null,
220     fine timestamp not null,
221     -- Vincolo di integrita': l'evento deve iniziare prima di
finire
222     check (inizio <= fine),
223     id_programma text references programma(id_programma) on delete
cascade not null
224 );
225
226 create table organizzatore_comitato(
227     id_organizzatore integer references organizzatore(
id_organizzatore) on delete cascade,
228     id_comitato integer references comitato(id_comitato) on delete
cascade,
229     -- Vincolo di integrita': un organizzatore puo' far parte di
un comitato una sola volta
230     unique (id_organizzatore, id_comitato)
231 );

```

## 3.2 Definizione dei trigger

### 3.2.1 Check\_Programma\_Entry

In un programma non devono esserci eventi, intervalli o interventi che si sovrappongono. Per questo motivo definiamo il trigger `check_programma_entry` che viene eseguito per ogni inserimento o aggiornamento nelle tabelle `INTERVENTO`, `INTERVALLO` ed `EVENTO`.

```

1  create or replace function check_programma()
2  returns trigger as $$
3  declare

```

```

4      inizio_evento timestamp;
5      fine_evento timestamp;
6      inizio_intervallo timestamp;
7      fine_intervallo timestamp;
8      inizio_intervento timestamp;
9      fine_intervento timestamp;
10     intervento_id integer;
11     intervallo_id integer;
12     evento_id integer;
13     interventi_cur cursor for
14         select id_intervento
15         from intervento
16         where id_programma = new.id_programma;
17     intervalli_cur cursor for
18         select id_intervallo
19         from intervallo
20         where id_programma = new.id_programma;
21     eventi_cur cursor for
22         select id_evento
23         from evento
24         where id_programma = new.id_programma;
25 begin
26     /* Controlliamo che il punto non si sovrapponga con
27      * ciascun punto gia' presente nel programma */
28     open interventi_cur;
29     loop
30         fetch interventi_cur into intervento_id;
31         exit when not found;
32         select inizio,fine into inizio_intervento,fine_intervento
33         from intervento
34         where id_intervento = intervento_id;
35         if (inizio_intervento <= new.inizio OR
36             fine_intervento >= new.fine) then
37             raise exception 'Impossibile inserire un punto del
38 programma in questo orario';
39         end if;
40     end loop;
41     close interventi_cur;
42     open intervalli_cur;
43     loop
44         fetch intervalli_cur into intervallo_id;
45         exit when not found;
46         select inizio,fine into inizio_intervallo,fine_intervallo
47         from intervallo
48         where id_intervallo = intervallo_id;
49         if (inizio_intervallo <= new.inizio OR fine_intervallo >=
new.fine) then
            raise exception 'Impossibile inserire un punto del
programma in questo orario';

```

```

50     end if;
51 end loop;
52 close intervalli_cur;
53 open eventi_cur;
54 loop
55     fetch eventi_cur into evento_id;
56     exit when not found;
57     select inizio,fine into inizio_evento,fine_evento
58     from evento
59     where id_evento = evento_id;
60     if (inizio_evento <= new.inizio OR fine_evento >= new.fine)
61     then
62         raise exception 'Impossibile inserire un punto del programma
63         in questo orario';
64     end if;
65     end loop;
66     close eventi_cur;
67 return new;
68 end;
69 $$
70 language plpgsql;
71 create trigger check_programma
72 before insert or update on intervento
73 for each row
74 execute function check_programma();
75
76 create trigger check_programma
77 before insert or update on intervallo
78 for each row
79 execute function check_programma();
80
81 create trigger check_programma
82 before insert or update on evento
83 for each row
84 execute function check_programma();

```

**Listato 3.1:** *check\_programma\_entry*

### 3.2.2 Check\_Data\_Intervento, Check\_Data\_Intervallo, Check\_Data\_Evento

Ogni volta che viene inserito o aggiornato un intervento, un intervallo o un evento bisogna controllare sempre che la data di inizio e di fine sia coerente con quella della sessione cui appartengono:

```

1 create or replace function check_data()
2 returns trigger as $$
3 declare
4     inizio_sessione timestamp;
5     fine_sessione timestamp;
6 begin
7     select inizio,fine into inizio_sessione,fine_sessione

```

```

8  from sessione
9  where id_sessione =
10      (select id_sessione
11       from programma
12       where id_programma = new.id_programma);
13
14  if (new.inizio < inizio_sessione OR new.fine > fine_sessione)
15      then
16          raise exception 'Gli orari non sono compatibili con quelli
17          della sessione';
18      end if;
19  return new;
20 end;
21 $$ language plpgsql;
22
23 create trigger check_data_intervento
24 before insert or update on intervento
25 for each row
26 execute function check_data();
27
28 create trigger check_data_intervallo
29 before insert or update on intervallo
30 for each row
31 execute function check_data();
32
33 create trigger check_data_evento
34 before insert or update on evento
35 for each row
36 execute function check_data();

```

**Listato 3.2:** *check\_data\_intervento*

### 3.2.3 Create\_Programma\_Sessione

Il trigger Create\_Programma\_Sessione viene attivato subito dopo aver inserito una nuova sessione ed effettua l'inserimento di un programma vuoto associato alla sessione.

```

1  create or replace function create_programma_sessione()
2  returns trigger as $$
3  begin
4      insert into programma(id_sessione) values (new.id_sessione);
5      return new;
6  end;
7  $$ language plpgsql;
8
9  create trigger create_programma_sessione
10 after insert on sessione
11 for each row
12 execute function create_programma_sessione();

```

**Listato 3.3:** *create\_programma\_sessione*



### 3.2.4 Check\_Sala\_Session

Quando inseriamo una sessione bisogna stare attenti che la chiave esterna della sala sia effettivamente una sala appartenente alla sede che ospita la conferenza della sessione in questione. Il trigger `check_sala_sessione` effettua quindi questo controllo prima di ciascun inserimento nella tabella `SESSIONE`:

```
1  create or replace function check_sala_sessione()
2  returns trigger as $$
3  declare
4      sede integer;
5      sala integer;
6  begin
7      select id_sede into sede
8      from conferenza
9      where id_conferenza = new.id_conferenza;
10
11     select id_sala into sala
12     from sala
13     where id_sala = new.id_sala;
14
15     if sala is null then -- Nulla da controllare
16         return new;
17     end if;
18
19     IF sala NOT IN (
20         SELECT id_sala
21         FROM sala
22         WHERE id_sede = sede
23     ) THEN
24         RAISE EXCEPTION 'La sala selezionata non appartiene alla
25         sede della conferenza';
26     END IF;
27
28     return new;
29 end;
30 $$ language plpgsql;
31
32 create trigger check_sala_sessione
33 before insert or update on sessione
34 for each row
35 execute function check_sala_sessione();
```

### 3.2.5 Check\_Data\_Session

Analogamente ai trigger 3.2.2 si definisce il trigger `check_data_sessione` che controlla che le date di inizio e di fine di ciascuna sessione siano coerenti con quelle della relativa conferenza:

```
1  create or replace function check_data_sessione()
2  returns trigger as $$
```

```

3  declare
4      inizio_conferenza timestamp;
5      fine_conferenza timestamp;
6  begin
7      select inizio, fine into inizio_conferenza, fine_conferenza
8      from conferenza
9      where id_conferenza = new.id_conferenza;
10
11     if (new.inizio < inizio_conferenza OR new.fine >
12     fine_conferenza) then
13         raise exception 'Gli orari non sono compatibili con quelli
14         della conferenza';
15     end if;
16     return new;
17 end;
18 $$ language plpgsql;
19
20 create trigger check_data_sessione
21 before insert or update on sessione
22 for each row
23 execute function check_data_sessione();

```

**Listato 3.4:** *check\_data\_sessione*

### 3.2.6 Check\_Coordinatore\_Sessione

Quando si specifica il coordinatore della sessione bisogna controllare che questi appartenga al comitato scientifico che è il gruppo di organizzatori che si occupano della gestione delle conferenze e delle sessioni:

```

1  create or replace function check_coordinatore_sessione()
2  returns trigger as $$
3  declare
4      id_comitato_scientifico_conferenza integer;
5  begin
6
7      select comitato_s into id_comitato_scientifico_conferenza
8      from conferenza c
9      where c.id_conferenza = new.id_conferenza;
10
11     if (new.id_coordinatore is not null) then
12         if (id_comitato_scientifico_conferenza not in
13             (select id_comitato from organizzatore_comitato
14              where id_organizzatore = new.id_coordinatore)) then
15             raise exception 'Il coordinatore della sessione deve
16             appartenere al comitato scientifico della conferenza';
17         end if;
18     end if;
19     return new;
20 end;
21 $$ language plpgsql;

```

```

21
22 create trigger check_coordinatore_sessione
23 before insert or update on sessione
24 for each row
25 execute function check_coordinatore_sessione();

```

**Listato 3.5:** *check\_coordinatore\_sessione*

### 3.2.7 Create\_Comitati\_Conferenza

Gli enti che organizzano le conferenze nominano due comitati per ogni conferenza che organizzano. Per questo motivo, ogni volta che viene inserita una nuova conferenza viene attivato il trigger `create_comitati_conferenza` che si occupa di creare due nuovi comitati di tipologica *scientifica* e *locale* e associarli alla nuova conferenza appena create:

```

1 create or replace function create_comitati_conferenza()
2 returns trigger as $$
3 declare
4     id_comitatoscientifico integer;
5     id_comitatolocale integer;
6 begin
7     insert into comitato(tipologia) values ('scientifico') returning
8         id_comitato into id_comitatoscientifico;
9     insert into comitato(tipologia) values ('locale') returning
10        id_comitato into id_comitatolocale;
11     update conferenza
12     set comitato_s = id_comitatoscientifico,
13     comitato_l = id_comitatolocale
14     where id_conferenza = new.id_conferenza;
15 return new;
16 end;
17 $$ language plpgsql;
18
19 create trigger create_comitati_conferenza
20 after insert on conferenza
21 for each row
22 execute function create_comitati_conferenza();

```

**Listato 3.6:** *create\_comitati\_conferenza*

### 3.2.8 Check\_Comitati\_Conferenza

Ogni volta che si aggiorna una conferenza bisogna controllare che le chiavi esterne dei due comitati si riferiscano sempre a comitati della tipologia richiesta:

```

1 create or replace function check_comitati_conferenza() returns
2     trigger as $$
3 declare
4     id_comitato_scientifico integer;
5     id_comitato_locale integer;
6 begin
7     select id_comitato into id_comitato_scientifico

```

```

7  from comitato
8  where id_comitato = new.comitato_s;
9
10 select id_comitato into id_comitato_locale
11 from comitato
12 where id_comitato = new.comitato_l;
13
14 IF id_comitato_scientifico IS NULL THEN
15 return new;
16 END IF;
17
18 IF id_comitato_locale IS NULL THEN
19 Return new;
20 END IF;
21
22 IF (select tipologia from comitato where id_comitato =
23     id_comitato_scientifico) <> 'scientifico' THEN
24 RAISE EXCEPTION 'Il comitato scientifico deve essere scientifico
25     ';
26 END IF;
27
28 IF (select tipologia from comitato where id_comitato =
29     id_comitato_locale) <> 'locale' THEN
30 RAISE EXCEPTION 'Il comitato locale deve essere locale';
31 END IF;
32
33 return new;
34 end;
35 $$ language plpgsql;
36
37 create trigger check_comitati_conferenza
38 before update on conferenza
39 for each row
40 execute function check_comitati_conferenza();

```

Listato 3.7: *check\_comitati\_conferenza*

### 3.2.9 Check\_Sala\_Sessione\_Unica

Una sala non può ospitare più di una sessione alla volta.

```

1  create or replace function check_sala_sessione_unica()
2  returns trigger as $$
3  declare
4      inizio_sessione timestamp;
5      fine_sessione timestamp;
6      sessioni cursor for
7          select id_sessione
8          from sessione
9          where id_sala = new.id_sala;
10     sessione_id integer;

```

```

11  begin
12
13  open sessioni;
14  loop
15      fetch sessioni into sessione_id;
16      exit when not found;
17      select inizio,fine into inizio_sessione,fine_sessione
18      from sessione
19      where id_sessione = sessione_id;
20      if (new.inizio >= inizio_sessione AND new.inizio <=
fine_sessione)
21      OR (new.fine >= inizio_sessione AND new.fine <= fine_sessione)
22      then
23          raise exception 'Impossibile ospitare due sessioni';
24      end if;
25  end loop;
26  close sessioni;
27  return new;
28  end;
29  $$ language plpgsql;
30
31  create trigger check_sala_sessione_unica
32  before insert or update on sessione
33  for each row
execute function check_sala_sessione_unica();

```

Listato 3.8: *Check\_sala\_sessione\_unica*

### 3.2.10 Check\_Organizzatore\_Comitato

Ogni volta che si inserisce un nuovo organizzatore all'interno di un comitato bisogna controllare che questo appartenga ad uno degli enti che organizzano la conferenza.

```

1  create or replace function check_organizzatore_comitato()
2  returns trigger as $$
3  declare
4      ente_id integer;
5  begin
6
7      select id_ente into ente_id
8      from organizzatore o
9      where o.id_organizzatore = new.id_organizzatore;
10
11  IF ente_id NOT IN (
12      SELECT id_ente
13      FROM ente_conferenza
14      WHERE id_conferenza IN (
15          SELECT id_conferenza
16          FROM conferenza
17          WHERE NEW.id_comitato
18          IN (id_comitato_scientifico, id_comitato_locale)

```

```

19     )
20 ) THEN
21     RAISE EXCEPTION 'L''organizzatore deve appartenere ad un ente
    che ha organizzato la conferenza';
22 END IF;
23 return new;
24 end;
25 $$ language plpgsql;
26
27 create trigger check_organizzatore_comitato
28 before insert or update on organizzatore_comitato
29 for each row
30 execute function check_organizzatore_comitato();

```

**Listato 3.9:** *Check\_organizzatori\_comitato*

### 3.2.11 Delete\_Sessioni\_Conferenza

Nel caso in cui si volesse modificare la data di inizio o di fine di una conferenza vengono automaticamente cancellate le sessioni che si trovano escluse dal nuovo intervallo di date.

```

1  create or replace function delete_sessioni_conferenza()
2  returns trigger as $$
3  declare
4  sessioni_cur cursor for
5      select id_sessione
6      from sessione
7      where id_conferenza = old.id_conferenza;
8  sessione_id integer;
9  begin
10     open sessioni_cur;
11     loop
12         fetch sessioni_cur into sessione_id;
13         exit when not found;
14         if (select inizio
15             from sessione
16             where id_sessione = sessione_id) < new.inizio
17         OR (select fine
18             from sessione
19             where id_sessione = sessione_id) > new.fine then
20             delete from sessione where id_sessione = sessione_id;
21         end if;
22     end loop;
23     close sessioni_cur;
24     return new;
25 end;
26 $$ language plpgsql;
27
28 create trigger delete_sessioni_conferenza
29 before update on conferenza
30 for each row

```

```
31 execute function delete_sessioni_conferenza();
```

Listato 3.10: *delete\_sessioni\_conferenza*

## 3.3 Funzioni e procedure

### 3.3.1 Show\_Conferenze\_By\_Date(date,date)

La funzione Show\_Conferenze\_By\_Date prende in ingresso due date e restituisce l'insieme di tutte le conferenze comprese tra queste:

```
1 create or replace function show_conference_by_date(dataI date ,
2   dataF date)
3 returns setof conferenza as $$
4 begin
5 return query
6 select * from conferenza
7 where inizio >= start and fine <= dataF;
8 end;
9 $$ language plpgsql;
```

### 3.3.2 Show\_Conferenze\_By\_Sede(integer)

La funzione Show\_Conferenze\_By\_Sede prende in ingresso la chiave primaria di una sede e restituisce l'insieme di tutte le conferenze ospitate in quella determinata sede:

```
1 create or replace function show_conferences_by_sede(sede int)
2 returns setof conferenza as $$
3 begin
4 return query
5 select * from conferenza
6 where id_sede = sede;
7 end;
8 $$ language plpgsql;
```

### 3.3.3 Show\_comitato\_scientifico(integer)

La funzione Show\_comitato\_scientifico prende in ingresso la chiave primaria di una conferenza e restituisce la lista di tutti i membri organizzatori appartenenti al comitato scientifico della conferenza:

```
1 create or replace function
2 show_comitato_scientifico(conferenza int)
3 returns setof organizzatore as $$
4 begin
5 return query
6 select * from organizzatore
7 where id_organizzatore in (
8   select id_organizzatore
9   from organizzatore_comitato
10  where id_comitato = (
```

```

11     select id_comitato_scientifico
12     from conferenza
13     where id_conferenza = conferenza));
14 end;
15 $$ language plpgsql;

```

### 3.3.4 Show\_comitato\_locale(integer)

La funzione Show\_comitato\_locale prende in ingresso la chiave primaria di una conferenza e restituisce la lista di tutti i membri organizzatori appartenenti al comitato locale della conferenza:

```

1 create or replace function show_comitato_locale(conferenza int)
2 returns setof organizzatore as $$
3 begin
4 return query
5 select * from organizzatore
6 where id_organizzatore in (
7     select id_organizzatore
8     from organizzatore_comitato
9     where id_comitato = (
10         select id_comitato_locale
11         from conferenza
12         where id_conferenza = conferenza));
13 end;
14 $$ language plpgsql;

```

### 3.3.5 Show\_Partecipanti(integer)

La funzione Show\_Partecipanti prende in ingresso la chiave primaria di una conferenza e restituisce tutti i dettagli dei partecipanti di *tutte le sessioni* della conferenza.

```

1 create or replace function show_partecipanti(conferenza int)
2 returns setof partecipante as $$
3 begin
4 return query
5 select * from partecipante
6 where id_partecipante in (
7     select id_partecipante
8     from partecipazione
9     where id_sessione in (
10         select id_sessione
11         from sessione
12         where id_conferenza = conferenza));
13 end;
14 $$ language plpgsql;

```

### 3.3.6 Show\_Sessioni(integer)

La funzione show\_sessioni prende in ingresso la chiave primaria di una conferenza e restituisce tutti i dettagli delle sessioni.



```

1  create or replace function show_sessioni(conferenza int)
2  returns setof sessione as $$
3  begin
4  return query
5  select * from sessione
6  where id_conferenza = conferenza
7  order by inizio;
8  end;
9  $$ language plpgsql;

```

### 3.3.7 Show\_interventi\_sessione(integer)

La funzione `show_interventi_sessione` prende in ingresso la chiave primaria di una sessione e mostra tutti gli interventi presenti nel programma di tale sessione:

```

1  create or replace function
2  show_interventi_sessione(sessione int)
3  returns table
4  (
5      titolo text,
6      inizio timestamp,
7      fine timestamp,
8      abstract text,
9      speaker text
10 ) as $$
11 declare
12     programma int;
13 begin
14     select id_programma into programma
15     from programma
16     where id_sessione = sessione;
17
18     select titolo,inizio,fine,abstract, s.nome || ' ' || s.cognome
19     as speaker
20     from intervento i join speaker s on i.id_speaker = s.
21     id_speaker
22     where i.id_programma = programma
23     order by inizio;
24 end;
25 $$ language plpgsql;

```

### 3.3.8 Show\_intervalli\_sessione(integer)

La funzione `show_intervalli_sessione` prende in ingresso la chiave primaria di una sessione e mostra tutti gli intervalli presenti nel programma di tale sessione:

```

1  create or replace function
2  show_intervalli_sessione(sessione int)
3  returns table
4  (

```

```

5      tipologia intervallo_st,
6      inizio timestamp,
7      fine timestamp
8  )
9  as $$
10 declare
11 programma int;
12 begin
13 select id_programma into programma
14 from programma
15 where id_sessione = sessione;
16
17 select tipologia,inizio,fine
18 from intervallo i
19 where id_programma = programma
20 order by inizio;
21 end;
22 $$
23 language plpgsql;

```

### 3.3.9 Show\_eventi\_sociali\_sessione(integer)

La funzione `show_eventi_sociali_sessione` prende in ingresso la chiave primaria di una sessione e mostra tutti gli intervalli presenti nel programma di tale sessione:

```

1 create or replace function
2 show_eventi_sociali_sessione(sessione int)
3 returns table
4 (id_evento text,
5  tipologia text,
6  inizio timestamp,
7  fine timestamp)
8 as $$
9 declare
10 programma int;
11 begin
12 select id_programma into programma
13 from programma
14 where id_sessione = sessione;
15
16 select id_evento,tipologia,inizio,fine
17 from evento
18 where id_programma = programma
19 order by inizio;
20 end;
21 $$ language plpgsql;

```

### 3.3.10 Show\_keynote\_sessione(integer)

La funzione `show_keynote_sessione` prende in ingresso la chiave primaria di una sessione e mostra i dettagli del keynote speaker, se presente:

```
1 create or replace function show_keynote_sessione(sessione int)
2 returns table(
3 id_speaker text,
4 nome text,
5 cognome text,
6 titolo text,
7 email text,
8 ente text)
9 as $$
10 declare
11 speaker_id text;
12 begin
13 select id_programma into programma
14 from programma
15 where id_sessione = sessione;
16
17 select id_keynote into speaker_id
18 from programma
19 where id_sessione = sessione;
20
21 if not found then
22     raise notice 'Keynote non presente';
23 else
24     select s.id_speaker,s.nome,s.cognome,s.titolo,s.email,e.nome
25     from speaker s join ente e on s.id_ente = e.id_ente
26     where s.id_speaker = speaker_id;
27 end if;
28 end;
29 $$ language plpgsql;
```

### 3.3.11 Show\_Programma(integer)

La funzione `Show_Programma` prende in ingresso la chiave primaria di una sessione e restituisce una tabella che mostra tutti gli appuntamenti in programma in ordine cronologico:

```
1 CREATE OR REPLACE FUNCTION
2 show_programma(sessione int)
3 RETURNS TABLE
4 (
5     id_entry text,
6     appuntamento text,
7     inizio timestamp,
8     fine timestamp,
9     descrizione text,
10    speaker text
11 )
```

```

12 AS $$
13 DECLARE
14     programma text;
15 BEGIN
16     SELECT id_programma INTO programma
17     FROM programma
18     WHERE id_sessione = sessione;
19
20     RETURN QUERY
21     SELECT *
22     FROM (
23         SELECT distinct i.id_intervento AS id_entry,
24         'intervento' AS appuntamento,
25         i.inizio,
26         i.fine,
27         i.abstract,
28         s.nome || ' ' || s.cognome AS speaker
29     FROM intervento i
30     JOIN speaker s ON i.id_speaker = s.id_speaker
31     WHERE i.id_programma = programma
32
33     UNION ALL
34
35     SELECT i2.id_intervallo AS id_entry,
36     'intervallo' AS appuntamento,
37     i2.inizio,
38     i2.fine,
39     tipologia::text as descrizione,
40     NULL
41     FROM intervallo i2
42     WHERE i2.id_programma = programma
43
44     UNION ALL
45
46     SELECT e.id_evento AS id_entry,
47     'evento' AS appuntamento,
48     e.inizio,
49     e.fine,
50     e.tipologia::text AS descrizione,
51     NULL
52     FROM evento e
53     WHERE e.id_programma = programma
54     ) AS subquery
55     ORDER BY inizio;
56 END;
57 $$ LANGUAGE plpgsql;

```

### 3.3.12 Add\_Intervento(text,text,text,int,interval)

```

1  create or replace procedure
2  add_intervento(titolo text, abstract text, speaker text,
3  sessione int,durata interval)
4  as $$
5  declare
6  programma text;
7  id_entry text;
8  query text;
9  category text;
10 fine_prev timestamp;
11 begin
12 select id_programma into programma
13 from programma
14 where id_sessione = sessione;
15
16 select id,appuntamento,max(fine) into id_entry,category,
17 fine_prev
18 from show_programma(sessione)
19 GROUP BY id, appuntamento;
20
21 if(fine_prev is null) then
22 fine_prev := (select inizio from sessione where id_sessione =
23 sessione);
24 end if;
25
26 insert into intervento(titolo,abstract,id_speaker,id_programma,
27 inizio,fine)
28 values (titolo,abstract,speaker,programma,fine_prev,fine_prev+
29 durata);
30 raise notice 'Inserimento completato';
31 exception
32 when others then
33 raise notice '%', sqlerrm;
34 end;
35 $$ language plpgsql;

```

### 3.3.13 Add\_Intervallo(text,int,interval)

```

1  create or replace procedure
2  add_intervallo(tipologia text,sessioneP int,durata interval)
3  as $$
4  declare
5  programma text;
6  id_entry text;
7  query text;
8  category text;
9  fine_prev timestamp;
10 begin
11 select id_programma into programma

```

```

12  from programma
13  where id_sessione = sessioneP;
14
15  select id,appuntamento, max(fine) into id_entry,category,
      fine_prev
16  from show_programma(sessioneP)
17  GROUP BY id, appuntamento;
18
19  if(fine_prev is null) then
20  fine_prev := (select inizio from sessione where id_sessione =
      sessioneP);
21  end if;
22
23  insert into intervallo(tipologia,id_programma,inizio,fine)
24  values (tipologia::intervallo_st, programma, fine_prev,
      fine_prev+durata);
25  raise notice 'Inserimento completato';
26  exception
27  when others then
28  raise notice '%', sqlerrm;
29  end;
30  $$ language plpgsql;

```

### 3.3.14 Add\_evento(text,int,interval)

```

1  create or replace procedure
2  add_evento(tipologia text, session int, durata interval)
3  as $$
4  declare
5  programma text;
6  id_entry text;
7  query text;
8  category text;
9  fine_prev timestamp;
10 begin
11 select id_programma into programma
12 from programma
13 where id_sessione = session;
14
15 select id, appuntamento, max(fine) into id_entry,category,
      fine_prev
16 from show_programma(session)
17 GROUP BY id, appuntamento;
18
19 if(fine_prev is null) then
20 fine_prev := (select inizio from sessione where id_sessione =
      session);
21 end if;
22

```

```

23 insert into evento(tipologia,id_programma,inizio,fine)
24 values (tipologia,programma,fine_prev,fine_prev+durata);
25 raise notice 'Inserimento completato';
26 exception
27 when others then
28 raise notice '%', sqlerrm;
29 end;
30 $$ language plpgsql;

```

### 3.3.15 Add\_Conferenza\_Details(text,timestamp,timestamp,integer,text)

```

1 CREATE OR REPLACE FUNCTION add_conferenza_details(nome text,
2 inizio timestamp, fine timestamp, sede integer, abstract text)
3 RETURNS integer AS $$
4 DECLARE
5 id integer;
6 BEGIN
7 INSERT INTO conferenza(titolo, inizio, fine, id_sede,
8 descrizione)
9 VALUES (nome, inizio, fine, sede, abstract)
10 RETURNING id_conferenza INTO id;
11 raise notice 'Inserimento completato';
12 RETURN id;
13 EXCEPTION
14 WHEN OTHERS THEN
15 RAISE NOTICE 'Errore nell''inserimento di una conferenza: %',
16 SQLERRM;
17 RETURN 0;
18 END;
19 $$ LANGUAGE plpgsql;

```

### 3.3.16 Add\_ente(integer, integer)

```

1 create or replace procedure
2 add_ente(ente integer, conferenza integer)
3 as $$
4 begin
5 insert into ente_conferenza(id_ente,id_conferenza)
6 values (ente,conferenza);
7 raise notice 'Inserimento completato';
8 exception
9 when others then
10 raise notice '%', sqlerrm;
11 end;
12 $$ language plpgsql;

```

### 3.3.17 Add\_Sponsorizzazione(integer,numeric,char(3),integer)

```

1  create or replace procedure
2  add_sponsorizzazione(sponsor integer, contributo numeric(1000,2)
3  , valuta char(3), conferenza integer)
4  as $$
5  begin
6  insert into sponsorizzazione(id_sponsor,contributo,valuta,
7  id_conferenza)
8  values (sponsor,contributo,valuta,conferenza);
9  raise notice 'Inserimento completato';
10 exception
11 when others then
12 raise notice '%', sqlerrm;
13 end;
14 $$ language plpgsql;

```

### 3.3.18 Add\_Sessione(text,timestamp,timestamp, integer,integer)

```

1  create or replace procedure
2  add_sessione(titolo text, inizio timestamp, fine timestamp, sala
3  integer, conferenza integer)
4  as $$
5  begin
6  insert into sessione(titolo,inizio,fine,id_sala,id_conferenza)
7  values (titolo,inizio,fine,sala,conferenza);
8  raise notice 'Inserimento completato';
9  exception
10 when others then
11 raise notice '%', sqlerrm;
12 end;
13 $$ language plpgsql;

```

### 3.3.19 Add\_Partecipante(integer, integer)

```

1  create or replace procedure
2  add_partecipante(partecipante integer, sessione integer)
3  as $$
4  begin
5  insert into partecipante_sessione(id_partecipante,id_sessione)
6  values (partecipante,sessione);
7  raise notice 'Inserimento completato';
8  exception
9  when others then
10 raise notice '%', sqlerrm;
11 end;
12 $$ language plpgsql;

```

### 3.3.20 Add\_Enti(integer,text)



```

1 CREATE OR REPLACE PROCEDURE
2 add_enti(conferenza integer, sigle text)
3 AS $$
4 DECLARE
5 sigla_ente text;
6 ente_id integer;
7 BEGIN
8 FOR sigla_ente IN SELECT unnest(string_to_array(sigle, ','))
9 LOOP
10 SELECT id_ente INTO ente_id FROM ente WHERE sigla = sigla_ente;
11 INSERT INTO ente_conferenza(id_ente, id_conferenza) VALUES (
12     ente_id, conferenza);
13 END LOOP;
14 RAISE NOTICE 'Inserimento completato';
15
16 EXCEPTION
17 WHEN OTHERS THEN
18 RAISE EXCEPTION 'Errore durante l''inserimento delle tuple nella
19     tabella ente_conferenza: %', SQLERRM;
20 END;
21 $$ LANGUAGE plpgsql;

```

### 3.3.21 Add\_Conferenza(text,timestamp,timestamp,integer, text, text)

```

1 create or replace procedure
2 add_conferenza(nome text, inizio timestamp, fine timestamp, sede
3     integer, descrizione text, sigle text)
4 as $$
5 declare
6 id_conferenza int;
7 begin
8 id_conferenza := add_conferenza_details(nome,inizio,fine,sede,
9     descrizione);
10 call add_enti(id_conferenza,sigle);
11 exception
12 when others then
13 raise notice '%', sqlerrm;
14 end;
15 $$ language plpgsql;

```

### 3.3.22 Slitta\_Conferenza(interval)

```

1 create or replace procedure
2 slitta_conferenza(id_conferenza integer, durata interval)
3 as $$
4 declare
5     id_sessione integer;

```

```

6      id_intervento text;
7      id_evento text;
8      id_intervallo text;
9      sessioni cursor for
10         select id_sessione
11         from sessione
12         where id_conferenza = id_conferenza;
13
14      interventi cursor for
15         select id_intervento
16         from intervento i join programma p
17         on i.id_programma = p.id_programma
18         where p.id_sessione in
19         (select id_sessione
20         from sessione
21         where id_conferenza = id_conferenza);
22
23      intervalli cursor for
24         select id_intervallo
25         from intervallo i join programma p
26         on i.id_programma = p.id_programma
27         where p.id_sessione in
28         (select id_sessione
29         from sessione
30         where id_conferenza = id_conferenza);
31
32      eventi cursor for
33         select id_evento
34         from evento e join programma p
35         on e.id_programma = p.id_programma
36         where p.id_sessione in
37         (select id_sessione
38         from sessione
39         where id_conferenza = id_conferenza);
40  begin
41
42      alter table conferenza disable trigger check_data_conferenza;
43      alter table sessione disable trigger check_data_sessione;
44      alter table interventi disable trigger check_data_intervento;
45      alter table intervallo disable trigger check_data_intervallo;
46      alter table evento disable trigger check_data_evento;
47      alter table conferenza disable trigger
48      delete_sessioni_conferenza;
49
50      update conferenza
51      set inizio = inizio + durata, fine = fine + durata
52      where id_conferenza = id_conferenza;
53
54      open sessioni;

```

```
54 loop
55     fetch sessioni into id_sessione;
56     exit when not found;
57
58     update sessione
59     set inizio = inizio + durata, fine = fine + durata
60     where id_sessione = id_sessione;
61
62     open interventi;
63     loop
64         fetch interventi into id_intervento;
65         exit when not found;
66
67         update intervento
68         set inizio = inizio + durata, fine = fine + durata
69         where id_intervento = id_intervento;
70     end loop;
71     close interventi;
72
73     open intervalli;
74     loop
75         fetch intervalli into id_intervallo;
76         exit when not found;
77
78         update intervallo
79         set inizio = inizio + durata, fine = fine + durata
80         where id_intervallo = id_intervallo;
81     end loop;
82     close intervalli;
83
84     open eventi;
85     loop
86         fetch eventi into id_evento;
87         exit when not found;
88
89         update evento
90         set inizio = inizio + durata, fine = fine + durata
91         where id_evento = id_evento;
92     end loop;
93     close eventi;
94 end loop;
95 close sessioni;
96 raise notice 'Slittamento completato';
97
98 alter table conferenza enable trigger check_data_conferenza;
99 alter table sessione enable trigger check_data_sessione;
100 alter table evento enable trigger check_data_evento;
101 alter table intervallo enable trigger check_data_intervallo;
102 alter table interventi enable trigger check_data_intervento;
```

```

103 exception
104     when others then
105         raise notice '%', sqlerrm;
106 end;
107 $$ language plpgsql;

```

### 3.3.23 Show\_members()

```

1  create or replace function
2  show_members(conferenza integer)
3  returns table
4  (
5  id integer,
6  nome text,
7  cognome text,
8  email text,
9  titolo titolo_st,
10 sigla varchar(7)
11 ) as $$
12 begin
13 return query
14 select o.id_organizzatore, o.nome, o.cognome, o.email, o.titolo,
15        e.sigla
16 from organizzatore o join ente_conferenza ec natural join ente e
17 on o.id_ente = ec.id_ente
18 where ec.id_conferenza = conferenza
19 GROUP by e.sigla;
20 end;
21 $$ language plpgsql;

```

### 3.3.24 Show\_percentage\_interventi(int,int)

```

1  create or replace function
2  show_percentage_interventi(mese int, anno int)
3  returns table
4  (
5      sigla varchar(7),
6      percentuale text
7  ) as $$
8  declare
9  totale int;
10 begin
11 select count(*) into totale
12 from intervento
13 where date_part('month',inizio) = mese and date_part('year',
14        inizio) = anno;
15
16 return query

```

```

16  select e.sigla, (count(*)*100/totale)::text || '%'
17  from intervento i join speaker s
18  on i.id_speaker = s.id_speaker join ente e
19  on s.id_ente = e.id_ente
20  where date_part('month',inizio) = mese and date_part('year',
    inizio) = anno
21  group by e.sigla;
22  end;
23  $$
24  language plpgsql;

```

### 3.3.25 Show\_percentage(int)

```

1  create or replace function
2  show_percentage_interventi(anno int)
3  returns table
4  (
5      sigla varchar(7),
6      percentuale text
7  ) as $$
8  declare
9      totale int;
10  begin
11      select count(*) into totale
12      from intervento
13      where date_part('year',inizio) = anno;
14
15      return query
16      select e.sigla, (count(*)*100/totale)::text || '%'
17      from intervento i join speaker s
18      on i.id_speaker = s.id_speaker join ente e
19      on s.id_ente = e.id_ente
20      where date_part('year',inizio) = anno
21      group by e.sigla;
22  end;
23  $$
24  language plpgsql;

```

## 3.4 Definizione delle viste

### 3.4.1 SediView

```

1  create view SediView as
2  select s.nome as Sede,
3  i.via || ', '
4  || i.civico
5  || ', '
6  || i.cap

```

```

7  || ', '
8  || i.city
9  || ' ('
10 || i.provincia
11 || '), '
12 || i.nazione as Indirizzo
13 from sede s natural join indirizzo i;

```

### 3.4.2 Conferenze\_Sede

```

1  create view conferenze_sede as
2  select s.nome as Sede, count(id_conferenza) as Numero_Conferenze
3  from sede s, conferenza c
4  where s.id_sede = c.id_sede
5  group by s.nome;

```

### 3.4.3 Interventi\_Speaker

```

1  create view interventi_speaker as
2  select s.nome || ' ' || s.cognome as Speaker, count(i.
   id_intervento)
3  from speaker s, intervento i
4  where s.id_speaker = i.id_speaker
5  group by s.nome, s.cognome;

```

### 3.4.4 Partecipanti\_Sessione

```

1  create view partecipanti_sessioni as
2  select s.titolo as Sessione,
3         count(p.id_partecipante) as Numero_partecipanti
4  from sessione s, partecipazione p
5  where s.id_sessione = p.id_sessione
6  group by s.titolo;

```

### 3.4.5 Partecipanti\_Conferenze

```

1  create view partecipanti_conferenze as
2  select c.titolo as Conferenza,
3         count(p.id_partecipante) as Numero_partecipanti
4  from conferenza c, sessione s, partecipazione p
5  where c.id_conferenza = s.id_conferenza
6  and s.id_sessione = p.id_sessione
7  group by c.titolo;
8

```

# Appendice A

## Dizionari

### A.1 Dizionario dei dati

Classe	Descrizione	Attributi
<b>Comitato</b>	Tabella che descrive i comitati che si occupano della logistica e della pianificazione delle conferenze scientifiche.	<b>id_comitato</b> ( <i>serial</i> ) ( <i>totale</i> ): Identificatore univoco per un comitato.  <b>tipologia</b> ( <i>comitato_st</i> )( <i>totale</i> ): Specifica il tipo di comitato (scientifico o locale).
<b>Conferenza</b>	Tabella che descrive le conferenze scientifiche.	<b>Id_Conferenza</b> ( <i>serial</i> )( <i>totale</i> ): Chiave primaria per una conferenza. <b>Titolo</b> ( <i>Text</i> ) ( <i>totale</i> ): Specifica il titolo della conferenza scientifica. <b>Descrizione</b> ( <i>Text</i> )( <i>parziale</i> ): Fornisce una descrizione della conferenza scientifica. <b>Inizio</b> ( <i>Timestamp</i> )( <i>totale</i> ): Indica l'inizio della conferenza. <b>Fine</b> ( <i>Timestamp</i> )( <i>totale</i> ) : Indica la fine della conferenza.
<b>Ente</b>	Tabella delle istituzioni	<b>Id_Ente</b> ( <i>serial</i> )( <i>totale</i> ): Identificatore primario di una istituzione. <b>Nome</b> ( <i>Text</i> )( <i>totale</i> ): Nome dell'istituzione. <b>Sigla</b> ( <i>Varchar(7)</i> )( <i>totale</i> ) : Sigla dell'istituzione.
<b>Evento</b>	Eventi sociali presenti all'interno di una conferenza.	<b>Id_Evento</b> ( <i>Serial</i> )( <i>Totale</i> ): Identificatore primario per un evento. <b>Tipologia</b> ( <i>text</i> )( <i>totale</i> ): Stringa descrittiva della tipologia dell'evento. <b>Inizio</b> ( <i>Timestamp</i> )( <i>totale</i> ): Indica l'inizio dell'evento.

*Continua nella prossima pagina*

Continua dalla pagina precedente

Classe	Descrizione	Attributi
		<b>Fine</b> ( <i>Timestamp</i> )( <i>totale</i> ) : Indica la fine dell'evento.
<b>Indirizzo</b>	Tabella degli indirizzi per ogni sede	<b>Id_Indirizzo</b> ( <i>serial</i> )( <i>totale</i> ): Chiave primaria. <b>Via</b> ( <i>text</i> )( <i>parziale</i> ): nome della via. <b>Civico</b> ( <i>text</i> )( <i>parziale</i> ): civico della sede. <b>Cap</b> ( <i>char</i> (5))( <i>parziale</i> ): codice di avviamento postale <b>Città</b> ( <i>text</i> )( <i>parziale</i> ): città della sede. <b>Provincia</b> ( <i>varchar</i> (2)): provincia della città. <b>Stato</b> ( <i>text</i> )( <i>parziale</i> ): stato della sede.
<b>Intervallo</b>	Descrittore degli intervalli presenti all'interno di una conferenza scientifica.	<b>Id_Intervallo</b> ( <i>Serial</i> )( <i>Totale</i> ): Identificatore primario per un evento.  <b>Tipologia</b> ( <i>Intervallo_ST</i> )( <i>totale</i> ): Specifica il tipo di intervallo (pranzo o coffee break). <b>Inizio</b> ( <i>Timestamp</i> )( <i>totale</i> ): Indica l'inizio dell'intervallo. <b>Fine</b> ( <i>Timestamp</i> )( <i>totale</i> ) : Indica la fine dell'intervallo.
<b>Intervento</b>	Descrittore degli interventi che si tengono all'interno delle sessioni.	<b>Id_Intervento</b> ( <i>Serial</i> )( <i>totale</i> ): Identificatore primario di un intervento.  <b>Titolo</b> ( <i>Text</i> ) ( <i>totale</i> ): Specifica il titolo dell'intervento. <b>Abstract</b> ( <i>Text</i> )( <i>parziale</i> ): Fornisce una descrizione dell'intervento. <b>Inizio</b> ( <i>Timestamp</i> )( <i>totale</i> ): Indica l'inizio dell'intervento. <b>Fine</b> ( <i>Timestamp</i> )( <i>totale</i> ) : Indica la fine dell'intervento.
<b>Organizzatore</b>	Descrittore dei membri dei comitati.	<b>Id_Organizzatore</b> ( <i>serial</i> )( <i>Totale</i> ): Identificatore principale di un organizzatore. <b>Nome</b> ( <i>text</i> )( <i>totale</i> ): nome dell'organizzatore. <b>Cognome</b> ( <i>text</i> )( <i>totale</i> ): cognome dell'organizzatore.

Continua nella prossima pagina



Continua dalla pagina precedente

Classe	Descrizione	Attributi
		<b>Titolo</b> ( <i>Titolo_ST</i> )( <i>parziale</i> ): Titolo accademico dell'organizzatore <b>Email</b> ( <i>Text</i> )( <i>Parziale</i> ): Email dell'organizzatore
<b>Partecipante</b>	Descrittore dei partecipanti delle sessioni.	<b>Id_Partecipante</b> ( <i>serial</i> )( <i>Totale</i> ): Identificatore principale di un partecipante. <b>Nome</b> ( <i>text</i> )( <i>totale</i> ): nome dell'organizzatore. <b>Cognome</b> ( <i>text</i> )( <i>totale</i> ): cognome dell'organizzatore. <b>Titolo</b> ( <i>Titolo_ST</i> )( <i>parziale</i> ): Titolo accademico del partecipante. <b>Email</b> ( <i>Text</i> )( <i>Parziale</i> ): Email del partecipante.
<b>Programma</b>	Tabella dei programmi delle sessioni.	<b>Id_Programma</b> ( <i>serial</i> )( <i>totale</i> ): Identificatore principale dei programmi.
<b>Sala</b>	Tabella delle sale di ciascuna sede.	<b>Id_sala</b> ( <i>serial</i> )( <i>totale</i> ): identificatore principale di ciascuna sala. <b>Nome</b> ( <i>Text</i> )( <i>totale</i> ): nome della sala. <b>Capienza</b> ( <i>int</i> )( <i>totale</i> ): capienza della sala.
<b>Sede</b>	Descrizione delle sedi che ospitano le conferenze	<b>Id_Sede</b> ( <i>Serial</i> )( <i>totale</i> ) : Identificatore principale delle sedi. <b>Nome</b> ( <i>Text</i> )( <i>totale</i> ): nome della sede.
<b>Sessione</b>	Tabella delle sessioni di ciascuna conferenza.	<b>Id_Sessione</b> ( <i>Serial</i> )( <i>total</i> ): Identificatore primario di una sessione. <b>Titolo</b> ( <i>Text</i> ) ( <i>totale</i> ): Specifica il titolo della sessione. <b>Inizio</b> ( <i>Timestamp</i> )( <i>totale</i> ): Indica l'inizio della sessione. <b>Fine</b> ( <i>Timestamp</i> )( <i>totale</i> ) : Indica la fine della sessione.
<b>Speaker</b>	Descrittore dei vari speaker delle sessioni.	<b>Id_Speaker</b> ( <i>serial</i> )( <i>Totale</i> ): Identificatore principale di uno speaker. <b>Nome</b> ( <i>text</i> )( <i>totale</i> ): nome dello speaker. <b>Cognome</b> ( <i>text</i> )( <i>totale</i> ): cognome dello speaker. <b>Titolo</b> ( <i>Titolo_ST</i> )( <i>parziale</i> ): Titolo accademico dello speaker.

Continua nella prossima pagina

Continua dalla pagina precedente

Classe	Descrizione	Attributi
		<b>Email</b> ( <i>Text</i> )( <i>Parziale</i> ): Email dello speaker.
<b>Sponsor</b>	Tabella degli sponsor	<b>Id_Sponsor</b> ( <i>serial</i> )( <i>totale</i> ): Identificatore primario di uno sponsor. <b>Nome</b> ( <i>Text</i> )( <i>totale</i> ): Nome dello sponsor.
<b>Valuta</b>	Tabella delle valute	<b>Iso</b> ( <i>Char</i> (3))( <i>totale</i> ): codice univoco internazionale delle valute. <b>Nome</b> ( <i>text</i> )( <i>totale</i> ): nome della valuta. <b>Simbolo</b> ( <i>char</i> (1))( <i>totale</i> ): simbolo della valuta.

## A.2 Dizionario delle associazioni

Associazione	Descrizione	Classi coinvolte
<b>Appartiene_A</b>	Rappresenta l'appartenenza di un organizzatore ad una precisa istituzione.	<b>Organizzatore [0..*]</b> : indica l'organizzatore che appartiene all'ente. <b>Ente [0..1]</b> ruolo <b>in</b> : indica l'ente al quale appartiene un organizzatore.
<b>Appartiene_A</b>	Rappresenta l'appartenenza di un partecipante ad una precisa istituzione.	<b>Organizzatore [0..*]</b> : indica il partecipante che appartiene ad un ente. <b>Ente [0..1]</b> ruolo <b>istituzione</b> : indica l'ente al quale appartiene un partecipante.
<b>Appartiene_A</b>	Rappresenta l'appartenenza di uno speaker ad una precisa istituzione.	<b>Organizzatore [0..*]</b> : indica lo speaker che appartiene all'ente. <b>Ente [0..1]</b> ruolo <b>istituzione</b> : indica l'ente al quale appartiene uno speaker.
<b>Comitato_Conferenza</b>	Ogni conferenza è legata ai comitati che ne gestiscono l'organizzazione.	<b>Comitati [2..2]</b> : indica i due comitati nominati per la conferenza. <b>Conferenza [1..1]</b> ruolo <b>di</b> : ogni comitato appartiene ad una sola conferenza.

Continua nella pagina successiva

Continua dalla pagina precedente

<b>Associazione</b>	<b>Descrizione</b>	<b>Classi coinvolte</b>
<b>Sponsorizzazione_Conferenza</b>	Ogni conferenza ha varie sponsorizzazioni da parte degli Sponsor che contribuiscono alle spese generali.	<b>Sponsor [0..*]</b>  <b>Conferenza [0..*]</b>
<b>Svolta_In</b>	Specifica l'ubicazione di una conferenza in una sede.	<b>Conferenza [0..*]</b>  <b>Sede [1..1]</b>
<b>Svolta_In</b>	Specifica l'ubicazione di una sessione in una sala.	<b>Sessione [0..*]</b>  <b>Sala [1..1]</b>
<b>Coordina</b>	Ogni sessione ha un coordinatore.	<b>Sessione [0..1]</b>  <b>Organizzatore [1..1]</b>
<b>Sessioni_Conferenza</b>	Ogni conferenza è composta da una o più sessioni.	<b>Conferenza [1..1]</b>  <b>Sessioni [0..*]</b>
<b>Sale_Sede</b>	Ogni sede è composta da una o più sedi.	<b>Sede [1..1]</b>  <b>Sala [1..*]</b>
<b>Programma_Sessione</b>	Ogni sessione ha un programma	<b>Sessione[1..1]</b>  <b>Programma [1..1]</b>
<b>Programma_Intervento</b>	Ogni programma è un composto di vari interventi	<b>Programma [1..1]</b>  <b>Intervento [0..*]</b>
<b>Programma_Intervallo</b>	Ogni programma è un composto di vari intervalli	<b>Programma [1..1]</b>  <b>Intervallo [0..*]</b>
<b>Programma_Evento</b>	Ogni programma è un composto di vari eventi sociali	<b>Programma [1..*]</b>  <b>Evento [0..*]</b>
<b>Partecipante_Sessione</b>	Ogni sessione ha vari partecipanti che partecipano a varie sessioni	<b>Sessione [0..*]</b>  <b>Partecipante [0..*]</b>
<b>Speaker_Intervento</b>	Ogni intervento ha un suo speaker che può effettuare vari interventi	<b>Intervento [0..*]</b>

Continua nella pagina successiva

*Continua dalla pagina precedente*

<b>Associazione</b>	<b>Descrizione</b>	<b>Classi coinvolte</b>
		<b>Speaker [1..1]</b>
<b>Membro_Comitato</b>	Ogni comitato è composto da vari organizzatori che appartengono a vari comitati	<b>Organizzatore [0..*]</b>  <b>Comitato [0..*]</b>

### **A.3 Dizionario dei vincoli**

<b>Vincolo</b>	<b>Tipo</b>	<b>Descrizione</b>
----------------	-------------	--------------------