

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

1	Traccia	6
1.1	Output attesi dal committente	6
2	Progettazione	7
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
3	Implementazione fisica	15
3.1	Definizione delle tabelle	15
3.2	Definizione dei trigger	19
3.2.1	Check_Programma	19
3.2.2	Check_Data_Intervento, Check_Data_Intervallo, Check_Data_Evento	20
3.2.3	Create_Programma_Sessione	21
3.2.4	Check_Sala_Sessione	21
3.2.5	Check_Data_Sessione	22
3.2.6	Check_Coordinatore_Sessione	22
3.2.7	Create_Comitati_Conferenza	23
3.2.8	Check_Comitati_Conferenza	24
3.2.9	Check_Sala_Sessione_Unica	24
3.2.10	Check_Organizzatore_Comitato	25
3.2.11	Delete_Sessioni_Conferenza	26
3.2.12	Check_Capienza	26
3.3	Funzioni e procedure	27
3.3.1	Show_Conferenze_By_Date(date,date)	27
3.3.2	Show_Conferenze_By_Sede(integer)	27
3.3.3	Show_comitato_scientifico(integer)	27
3.3.4	Show_comitato_locale(integer)	28
3.3.5	Show_Partecipanti(integer)	28
3.3.6	Show_Sessioni(integer)	28
3.3.7	Show_interventi_sessione(integer)	29

3.3.8	Show_intervalli_sessione(integer)	29
3.3.9	Show_eventi_sociali_sessione(integer)	30
3.3.10	Show_keynote_sessione(integer)	30
3.3.11	Show_Programma(integer)	31
3.3.12	Add_Intervento(text,text,text,int,interval)	32
3.3.13	Add_Intervallo(text,int,interval)	33
3.3.14	Add_Evento(text,int,interval)	33
3.3.15	Add_Conferenza_Details(text,timestamp,timestamp,integer,text) .	34
3.3.16	Add_ente(integer, integer)	34
3.3.17	Add_Sponsorizzazione(integer,numeric,char(3),integer)	35
3.3.18	Add_Sessioni(text,timestamp,timestamp, integer,integer)	35
3.3.19	Add_Partecipante(integer, integer)	35
3.3.20	Add_Enti(integer,text)	36
3.3.21	Add_Conferenza(text,timestamp,timestamp,integer, text, text) . .	36
3.3.22	Slitta_Conferenza(interval)	36
3.3.23	Show_members()	38
3.3.24	Show_percentage_interventi(int,int)	39
3.3.25	Show_percentage(int)	39
3.4	Definizione delle viste	39
3.4.1	SediView	39
3.4.2	Conferenze_Sede	40
3.4.3	Interventi_Speaker	40
3.4.4	Partecipanti_Sessioni	40
3.4.5	Partecipanti_Conferenze	40
3.4.6	Sessioni	40
A	Dizionari	42
A.1	Dizionario dei dati	42
A.2	Dizionario delle associazioni	45
A.3	Dizionario dei vincoli	47

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	19
3.2 check_data_intervento	20
3.3 create_programma_sessione	21
3.4 check_data_sessione	22
3.5 check_coordinatore_sessione	22
3.6 create_comitati_conferenza	23
3.7 check_comitati_conferenza	24
3.8 Check_sala_sessione_unica	24
3.9 Check_organizzatori_comitato	25
3.10 delete_sessioni_conferenza	26

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 appartenenza.

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
Conferenza	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> .
Ente	Per gli enti che organizzano le conferenze scientifiche. Di ogni ente si conserva il <i>nome</i> e la <i>sigla</i> .
Sponsor	Per gli sponsor che coprono le spese della conferenza. Di ogni sponsor si conserva il <i>nome</i> .
Comitato	Per i gruppi di organizzatori che si occupano della gestione della conferenza. Si distinguono in comitati <i>scientifici</i> e <i>locali</i> .
Organizzatore	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> .
Sede	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> .
Sala	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> .
Sessione	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> .
Programma	Per il programma di ciascuna sessione. Ogni programma specifica la presenza di un <i>keynote speaker</i> , ovvero un partecipante di rilievo.
Intervento	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.
Partecipante	Per i partecipanti delle varie sessioni. Ogni partecipante ha gli stessi attributi degli organizzatori.
Intervallo	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> .
Evento sociale	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 4217¹.

¹ISO 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>
------------------	-----------	--------	------	---------------------

2. 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>
-------------------	------	---------	--------	-------	----------------

3. 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>
----------------------	-------------------	--------	----------	--------	------	---------------------

4. Una sala può ospitare più sessioni:

SESSIONE

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

5. Una sede può ospitare più conferenze:

CONFERENZA

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

6. 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:

1. 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>
--------------------	------	--------	------	----------------	----------------------	------------------------

2. 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>
---------------------	--------------------	-------------------

3. 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.

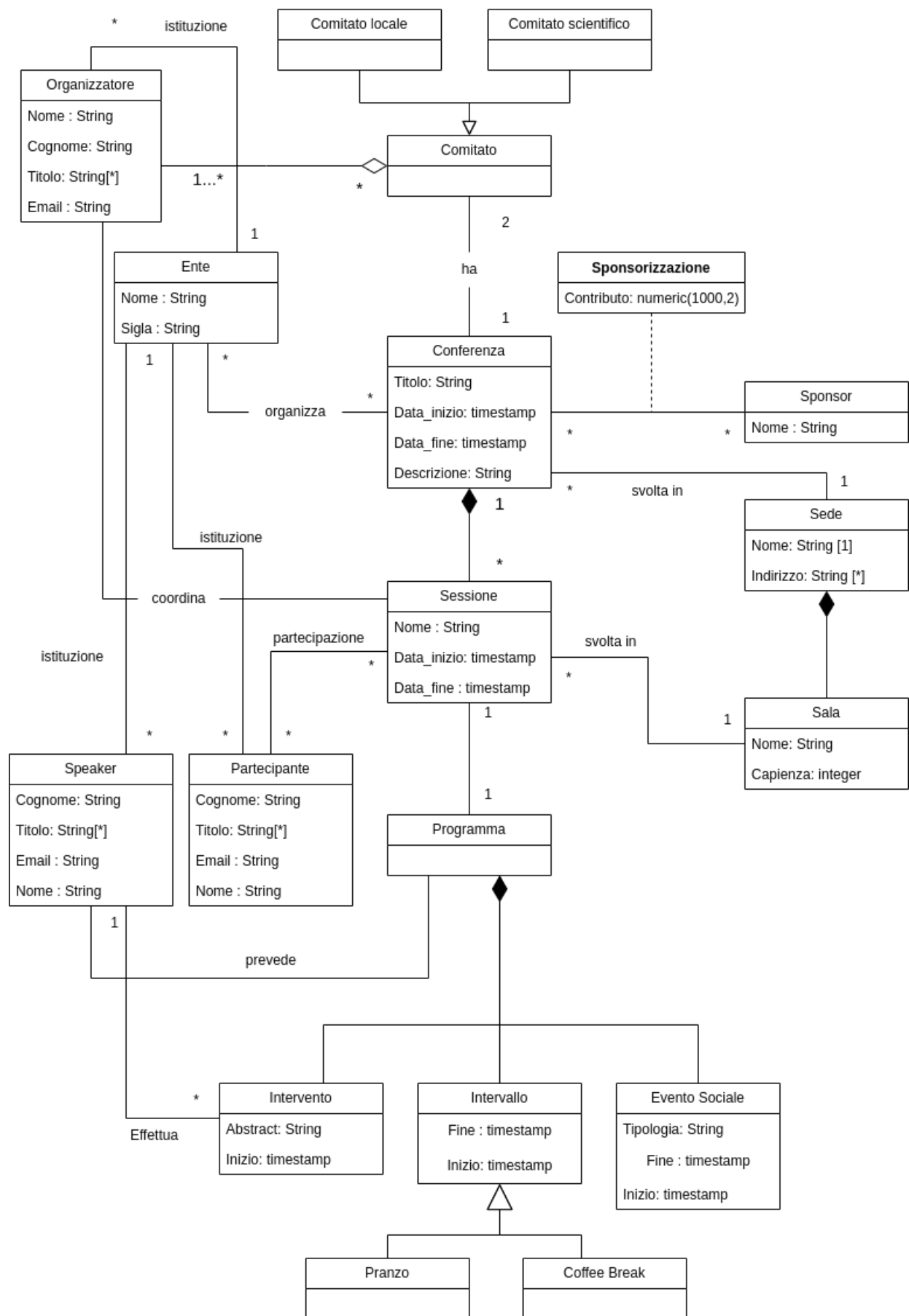


Figura 2.1: Schema concettuale del problema

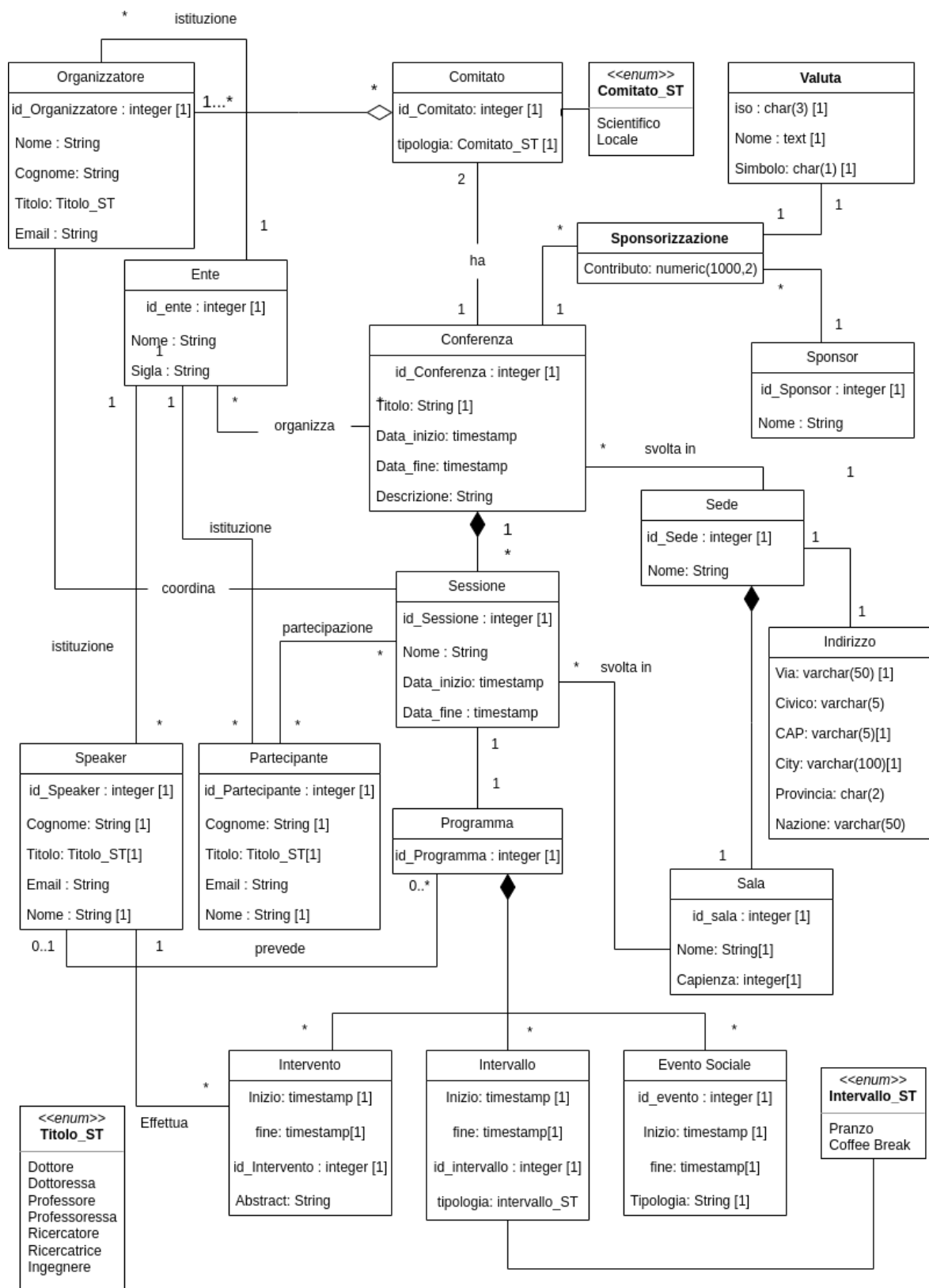
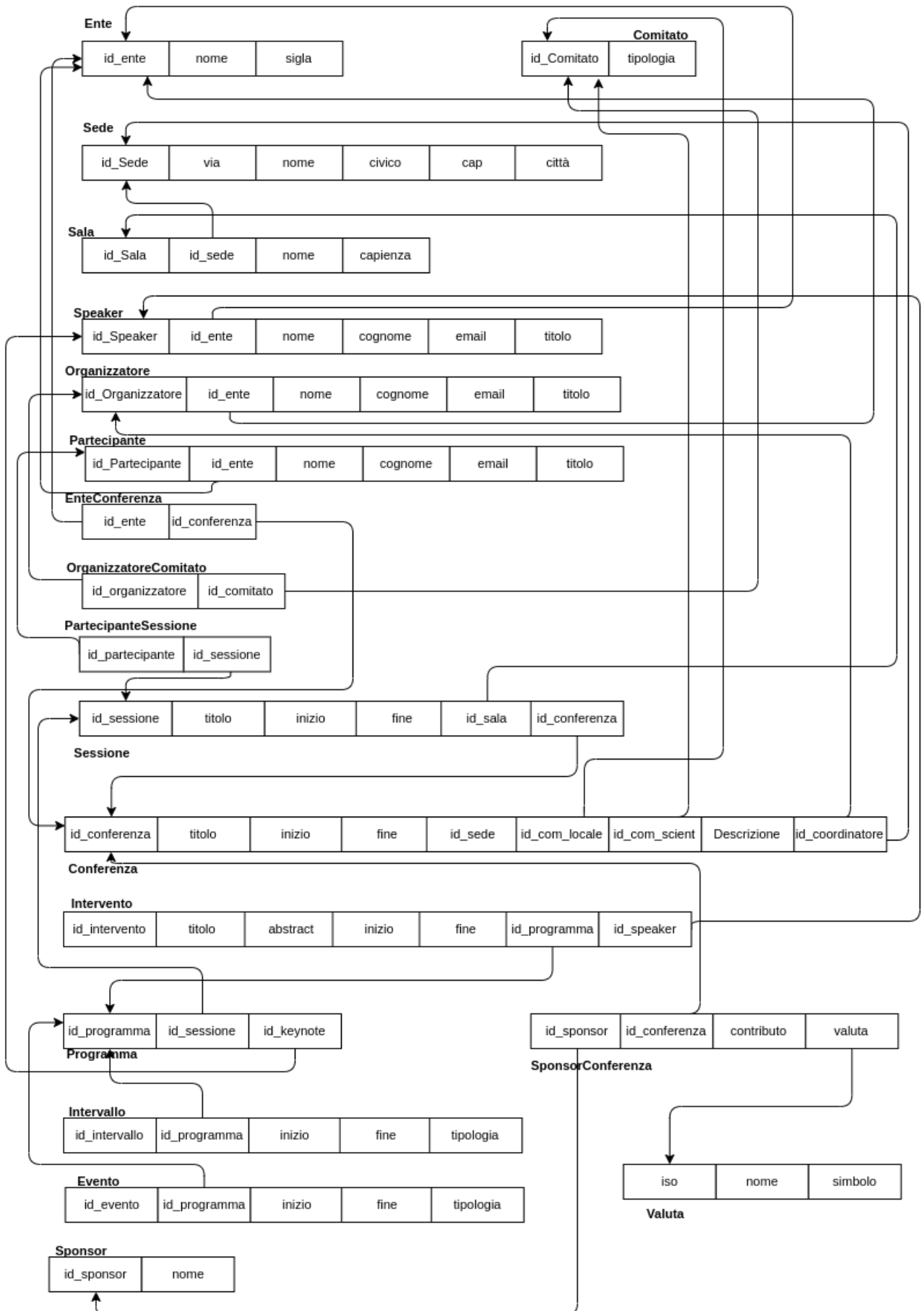


Figura 2.2: *Ristrutturazione dello schema concettuale*

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 delete set null
33 );
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','Dottorressa','Professore','
    Professoressa','Assistente','Ricercatore','Ricercatrice','Ingegnere');
50 create table organizzatore(
51     id_organizzatore serial primary key,
52     nome text not null,
53     cognome text not null,
54     titolo titolo_st,
55     email text not null unique,
56     id_ente text references ente(id_ente) on delete cascade
57 );
58
59
60 create table sala(
61     id_sala serial primary key,
62     nome text not null,
63     capienza integer not null,
64     id_sede integer references sede(id_sede) on delete cascade
65 );
66
67
68 create table conferenza(
69     id_conferenza serial primary key,
70     titolo text not null,
71     descrizione text not null,
72     inizio timestamp not null,
73     fine timestamp not null,
74     id_sede integer references sede(id_sede) on delete set null,
75     comitato_s integer references comitato(id_comitato) on delete set null,
76     comitato_l integer references comitato(id_comitato) on delete set null,
77     -- Vincolo di integrita': la conferenza deve iniziare prima di 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(id_organizzatore) on
    delete set null,
100    id_conferenza integer references conferenza(id_conferenza) on delete
    cascade,

```

```

101     id_sala integer references sala(id_sala) on delete set null,
102     -- Vincolo di integrita': la sessione deve iniziare prima di finire
103     check (inizio <= fine)
104 );
105
106 create table partecipazione(
107     id_partecipante integer references partecipante(id_partecipante) on delete
108     cascade,
109     id_sessione integer references sessione(id_sessione) on delete cascade,
110     -- Vincolo di unicity: un partecipante puo' partecipare ad una sessione
111     una sola volta
112     unique (id_partecipante,id_sessione)
113 );
114
115 create table ente_conferenza(
116     id_ente text references ente(id_ente) on delete cascade,
117     id_conferenza integer references conferenza(id_conferenza) on delete
118     cascade,
119     -- Vincolo di unicity: un ente puo' organizzare una conferenza una sola
120     volta
121     unique (id_ente,id_conferenza)
122 );
123
124 create table valuta(
125     iso char(3) primary key,
126     nome text not null,
127     simbolo text not null
128 );
129
130 create table sponsor_conferenza(
131     id_sponsor integer references sponsor(id_sponsor) on delete cascade not
132     null,
133     contributo numeric(1000,2) not null,
134     valuta char(3) references valuta(iso) not null,
135     id_conferenza integer references conferenza(id_conferenza) on delete
136     cascade not null,
137     -- Vincolo di unicity: uno sponsor puo' sponsorizzare una conferenza una
138     sola volta
139     unique (id_sponsor,id_conferenza)
140 );
141
142 create sequence speaker_id_speaker_seq;
143 create or replace function nextval_speaker() returns text as $$
144 begin
145     return 'SPK' || to_char(nextval('speaker_id_speaker_seq'),'FM0000');
146 end;
147 $$
148 language plpgsql;
149
150 create table speaker(
151     id_speaker text primary key default nextval_speaker(),
152     nome text not null,
153     cognome text not null,
154     titolo titolo_st,
155     email text not null unique,
156     id_ente text references ente(id_ente) on delete cascade NOT NULL
157 );
158
159 create sequence programma_id_programma_seq;
160 create or replace function nextval_programma() returns text as $$
161 begin

```

```

155     return 'PRG' || to_char(nextval('programma_id_programma_seq'), 'FM0000');
156 end;
157 $$
158 language plpgsql;
159
160 create table programma(
161     id_programma text primary key default nextval_programma(),
162     id_sessione integer references sessione(id_sessione) on delete cascade not
163     null,
164     id_keynote text references speaker(id_speaker) on delete set null,
165     unique (id_programma, id_sessione)
166 );
167
168 create sequence intervento_id_intervento_seq;
169 create or replace function nextval_intervento() returns text as $$
170 begin
171     return 'INT' || to_char(nextval('intervento_id_intervento_seq'), 'FM0000');
172 end;
173 $$
174 language plpgsql;
175
176 create table intervento(
177     id_intervento text primary key default nextval_intervento(),
178     titolo text not null,
179     abstract text not null,
180     inizio timestamp not null,
181     fine timestamp not null,
182     id_speaker text references speaker(id_speaker) on delete cascade,
183     id_programma text references programma(id_programma) on delete cascade not
184     null,
185     -- Vincolo di unicita': uno speaker puo' intervenire in una sessione una
186     -- Vincolo di integrita': l'intervento deve iniziare prima di finire
187     unique (id_speaker, id_programma),
188     check (inizio <= fine)
189 );
190
191 create type intervallo_st as enum ('pranzo', 'coffee break');
192 create sequence intervallo_id_intervallo_seq;
193 create or replace function nextval_intervallo() returns text as $$
194 begin
195     return 'BRK' || to_char(nextval('intervallo_id_intervallo_seq'), 'FM0000');
196 end;
197 $$
198 language plpgsql;
199
200 create table intervallo(
201     id_intervallo text primary key default nextval_intervallo(),
202     tipologia intervallo_st not null,
203     inizio timestamp not null,
204     fine timestamp not null,
205     -- Vincolo di integrita': l'intervallo deve iniziare prima di finire
206     check (inizio <= fine),
207     id_programma text references programma(id_programma) on delete cascade not
208     null
209 );
210
211 create sequence evento_id_evento_seq;
212 create or replace function nextval_evento() returns text as $$
213 begin
214     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
224     null
225 );
226
227 create table organizzatore_comitato(
228     id_organizzatore integer references organizzatore(id_organizzatore) on
229     delete cascade,
230     id_comitato integer references comitato(id_comitato) on delete cascade,
231     -- Vincolo di integrita': un organizzatore puo' far parte di un comitato
232     una sola volta
233     unique (id_organizzatore,id_comitato)
234 );

```

3.2 Definizione dei trigger

3.2.1 Check_Programma

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() returns trigger as $$
2 declare
3     inizio_evento timestamp;
4     fine_evento timestamp;
5     inizio_intervallo timestamp;
6     fine_intervallo timestamp;
7     inizio_intervento timestamp;
8     fine_intervento timestamp;
9     intervento_id text;
10    intervallo_id text;
11    evento_id text;
12    interventi_cur cursor for select id_intervento from intervento where
13        id_programma = new.id_programma;
14    intervalli_cur cursor for select id_intervallo from intervallo where
15        id_programma = new.id_programma;
16    eventi_cur cursor for select id_evento from evento where id_programma = new.
17        id_programma;
18 begin
19     open interventi_cur;
20     loop
21         fetch interventi_cur into intervento_id;
22         exit when not found;
23         select inizio,fine into inizio_intervento,fine_intervento
24         from intervento
25         where id_intervento = intervento_id;
26         if (new.inizio>=inizio_intervento AND new.fine<=fine_intervento) then
27             raise exception 'Impossibile inserire questo intervento in questo orario';
28         end if;
29     end loop;

```

```

27 close interventi_cur;
28 open intervalli_cur;
29 loop
30 fetch intervalli_cur into intervallo_id;
31 exit when not found;
32 select inizio,fine into inizio_intervallo,fine_intervallo
33 from intervallo
34 where id_intervallo = intervallo_id;
35 if (new.inizio>=inizio_intervallo AND new.fine<=fine_intervallo) then
36 raise exception 'Impossibile inserire questo intervallo in questo orario';
37 end if;
38 end loop;
39 close intervalli_cur;
40 open eventi_cur;
41 loop
42 fetch eventi_cur into evento_id;
43 exit when not found;
44 select inizio,fine into inizio_evento,fine_evento
45 from evento
46 where id_evento = evento_id;
47 if (new.inizio>=inizio_evento AND new.fine<=fine_evento) then
48 raise exception 'Impossibile inserire questo evento in questo orario';
49 end if;
50 end loop;
51 close eventi_cur;
52 return new;
53 end;
54 $$ language plpgsql;

```

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

```

```

24 execute function check_data();
25
26 create trigger check_data_intervallo
27 before insert or update on intervallo
28 for each row
29 execute function check_data();
30
31 create trigger check_data_evento
32 before insert or update on evento
33 for each row
34 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_Sessione

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 sede della
conferenza';
25     END IF;
26
27     return new;
28 end;
29 $$ language plpgsql;
30
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_Sessione

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 $$
3  declare
4      inizio_conferenza timestamp;
5      fine_conferenza timestamp;
6  begin
7      select inizio, fine into inizio_conferenza, fine_conferenza
8      from conferenza
9      where id_conferenza = new.id_conferenza;
10
11     if (new.inizio < inizio_conferenza OR new.fine > fine_conferenza) then
12         raise exception 'Gli orari non sono compatibili con quelli della
conferenza';
13     end if;
14     return new;
15 end;
16 $$ language plpgsql;
17
18 create trigger check_data_sessione
19 before insert or update on sessione
20 for each row
21 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 appartenere al
16         comitato scientifico della conferenza';
17     end if;
18 end if;
19 return new;
20 end;
21 $$ language plpgsql;
22
23 create trigger check_coordinatore_sessione
24 before insert or update on sessione
25 for each row
26 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 $$
3 declare
4     id_comitatoscientifico integer;
5     id_comitatolocale integer;
6 begin
7     insert into comitato(tipologia) values ('scientifico') returning id_comitato
8     into id_comitatoscientifico;
9     insert into comitato(tipologia) values ('locale') returning id_comitato into
10    id_comitatolocale;
11    update conferenza
12    set comitato_s = id_comitatoscientifico,
13    comitato_l = id_comitatolocale
14    where id_conferenza = new.id_conferenza;
15 return new;
16 end;
17 $$ 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 trigger as $$
2  declare
3  id_comitato_scientifico integer;
4  id_comitato_locale integer;
5  begin
6  select id_comitato into id_comitato_scientifico
7  from comitato
8  where id_comitato = new.comitato_s;
9
10 select id_comitato into id_comitato_locale
11 from comitato
12 where id_comitato = new.comitato_l;
13
14 IF id_comitato_scientifico IS NULL THEN
15 return new;
16 END IF;
17
18 IF id_comitato_locale IS NULL THEN
19 Return new;
20 END IF;
21
22 IF (select tipologia from comitato where id_comitato =
23     id_comitato_scientifico) <> 'scientifico' THEN
24 RAISE EXCEPTION 'Il comitato scientifico deve essere scientifico';
25 END IF;
26
27 IF (select tipologia from comitato where id_comitato = id_comitato_locale) <>
28     'locale' THEN
29 RAISE EXCEPTION 'Il comitato locale deve essere locale';
30 END IF;
31
32 return new;
33 end;
34 $$ language plpgsql;
35
36 create trigger check_comitati_conferenza
37 before update on conferenza
38 for each row
39 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) then
22          raise exception 'Impossibile ospitare due sessioni';
23      end if;
24  end loop;
25  close sessioni;
26  return new;
27  end;
28  $$ language plpgsql;
29
30  create trigger check_sala_sessione_unica
31  before insert or update on sessione
32  for each row
33  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 $$
3  declare
4      ente_id integer;
5  begin
6
7      select id_ente into ente_id
8      from organizzatore o
9      where o.id_organizzatore = new.id_organizzatore;
10
11  IF ente_id NOT IN (
12      SELECT id_ente
13      FROM ente_conferenza
14      WHERE id_conferenza IN (
15          SELECT id_conferenza
16          FROM conferenza
17          WHERE NEW.id_comitato
18          IN (id_comitato_scientifico, id_comitato_locale)
19      )
20  ) THEN
21      RAISE EXCEPTION 'L''organizzatore deve appartenere ad un ente che ha
22      organizzato la conferenza';
23  END IF;
24  return new;
25  end;
26  $$ language plpgsql;
27
28  create trigger check_organizzatore_comitato
29  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*

3.2.12 Check_Capienza

Ogni volta che si aggiunge un nuovo partecipante della sessione bisogna controllare prima che la capienza della sala dove si svolge la sessione non sia stata raggiunta:

```

1  create or replace function check_capienza_sala()
2  returns trigger as $$
3  declare
4  capienza_s integer;
5  partecipanti integer;
6  begin
7      select capienza into capienza_s
8      from sala
9      where id_sala = new.id_sala;
10
11     select count(*) into partecipanti

```

```

12  from partecipante
13  where id_sessione = new.id_sessione;
14
15  if (partecipanti >= capienza_s) then
16  raise exception 'La capienza della sala e'' stata raggiunta';
17  end if;
18  return new;
19  end;
20  $$ language plpgsql;
21
22  create trigger check_capienza_sala
23  before insert on partecipazione
24  for each row
25  execute function check_capienza_sala();
26

```

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, dataF date)
2  returns setof conferenza as $$
3  begin
4  return query
5  select * from conferenza
6  where inizio >= start and fine <= dataF;
7  end;
8  $$ 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 $$
11 declare
12     programma int;
13 begin
14     select id_programma into programma
15     from programma
16     where id_sessione = sessione;
17
18     select titolo,inizio,fine,abstract, s.nome || ' ' || s.cognome as speaker
19     from intervento i join speaker s on i.id_speaker = s.id_speaker
20     where i.id_programma = programma
21     order by inizio;
22 end;
23 $$ 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 $$
58 LANGUAGE plpgsql;

```

3.3.12 Add_Intervento(text,text,text,int,interval)

La procedura `Add_intervento` provvede all'inserimento di un intervento all'interno del programma della sessione. Questa calcola l'orario esatto in cui inserire il nuovo punto sulla base dell'ultimo punto in programma. Se non esistono punti in programma allora l'ora di inizio è calcolato come l'inizio della sessione:

```

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

```

3.3.13 Add_Intervallo(text,int,interval)

La procedura Add_Intervallo provvede all'inserimento di un intervallo all'interno del programma della sessione. Questa calcola l'orario esatto in cui inserire il nuovo punto sulla base dell'ultimo punto in programma. Se non esistono punti in programma allora l'ora di inizio è calcolato come l'inizio della sessione:

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

3.3.14 Add_Evento(text,int,interval)

La procedura Add_Evento provvede all'inserimento di un evento all'interno del programma della sessione. Questa calcola l'orario esatto in cui inserire il nuovo punto sulla base dell'ultimo punto in programma. Se non esistono punti in programma allora l'ora di inizio è calcolato come l'inizio della sessione:

```
1 create or replace procedure
2 add_evento
3 (tipologia text,
4 sessione_id int,
5 durata interval)
6 as $$
7 declare
8 programma_id text;
9 id text;
10 query text;
11 category text;
12 fine_prev timestamp;
13 begin
```

```

14 -- Recupera l'id del programma della sessione
15 select id_programma into programma_id
16 from programma
17 where id_sessione = sessione_id;
18
19 -- Recupera l'id dell'ultimo punto in programma, la tipologia e la fine
20 select max(fine) into fine_prev
21 from show_programma(sessione_id);
22
23 if (fine_prev is null) then
24 select inizio into fine_prev
25 from sessione
26 where id_sessione = sessione_id;
27 end if;
28
29 insert into evento(tipologia, id_programma, inizio, fine)
30 values (tipologia, programma_id, fine_prev, fine_prev+durata);
31 raise notice 'Inserimento completato';
32 exception
33 when others then
34 raise notice '%', sqlerrm;
35 end;
36 $$
37 language plpgsql;

```

3.3.15 Add_Conferenza_Details(text,timestamp,timestamp,integer,text)

La funzione Add_Conferenza_Details aggiunge una conferenza e restituisce la chiave primaria della nuova conferenza.

```

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

```

3.3.16 Add_ente(integer, integer)

La procedura Add_ente provvede all'inserimento di una nuova istituzione tra gli organizzatori di una conferenza.

```

1  create or replace procedure
2  add_ente(ente text, 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 $$
13 language plpgsql;

```

3.3.17 Add_Sponsorizzazione(integer,numeric,char(3),integer)

La procedura Add_Sponsorizzazione inserisce una nuova sponsorizzazione per la conferenza:

```

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

```

3.3.18 Add_Sessione(text,timestamp,timestamp, integer,integer)

La procedura Add_Sessione aggiunge una nuova sessione per la conferenza:

```

1  create or replace procedure
2  add_sessione(titolo text, inizio timestamp, fine timestamp, sala integer,
3  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)

La procedura Add_Partecipante inserisce un nuovo partecipante alla sessione:

```

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 $$
13 language plpgsql;

```

3.3.20 Add_Enti(integer,text)

```

1 CREATE OR REPLACE PROCEDURE
2 add_enti(conferenza integer, sigle text)
3 AS $$
4 DECLARE
5 sigla_ente text;
6 ente_id integer;
7 BEGIN
8 FOR sigla_ente IN SELECT unnest(string_to_array(sigle, ',')) LOOP
9 SELECT id_ente INTO ente_id FROM ente WHERE sigla = sigla_ente;
10
11 INSERT INTO ente_conferenza(id_ente, id_conferenza) VALUES (ente_id,
12 conferenza);
13 END LOOP;
14 RAISE NOTICE 'Inserimento completato';
15
16 EXCEPTION
17 WHEN OTHERS THEN
18 RAISE EXCEPTION 'Errore durante l''inserimento delle tuple nella tabella
19 ente_conferenza: %', SQLERRM;
20 END;
21 $$ 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 integer,
3 descrizione text, sigle text)
4 as $$
5 declare
6 id_conferenza int;
7 begin
8 id_conferenza := add_conferenza_details(nome,inizio,fine,sede,descrizione);
9 call add_enti(id_conferenza,sigle);
10 exception
11 when others then
12 raise notice '%', sqlerrm;
13 end;
14 $$ language plpgsql;

```

3.3.22 Slitta_Conferenza(interval)

```

1 create or replace procedure
2 slitta_conferenza(conferenza_id integer, durata interval)
3 as $$
4 declare
5 sessione_id integer;
6 intervento_id text;
7 evento_id text;
8 intervallo_id text;
9 sessioni cursor for
10 select id_sessione

```

```

11 from sessione
12 where id_conferenza = conferenza_id;
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 = conferenza_id);
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 = conferenza_id);
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 = conferenza_id);
40 begin
41 alter table conferenza disable trigger all;
42 alter table sessione disable trigger all;
43 alter table intervento disable trigger all;
44 alter table intervallo disable trigger all;
45 alter table evento disable trigger all;
46 alter table programma disable trigger all;
47 update conferenza
48 set inizio = inizio + durata, fine = fine + durata
49 where id_conferenza = conferenza_id;
50
51 open sessioni;
52 loop
53 fetch sessioni into sessione_id;
54 exit when not found;
55
56 update sessione
57 set inizio = inizio + durata, fine = fine + durata
58 where id_sessione = sessione_id;
59
60 open interventi;
61 loop
62 fetch interventi into intervento_id;
63 exit when not found;
64
65 update intervento
66 set inizio = inizio + durata, fine = fine + durata
67 where id_intervento = intervento_id ;
68 end loop;
69 close interventi;
70
71 open intervalli;

```

```

72 loop
73 fetch intervalli into intervallo_id;
74 exit when not found;
75
76 update intervallo
77 set inizio = inizio + durata, fine = fine + durata
78 where id_intervallo = intervallo_id;
79 end loop;
80 close intervalli;
81
82 open eventi;
83 loop
84 fetch eventi into evento_id;
85 exit when not found;
86
87 update evento
88 set inizio = inizio + durata, fine = fine + durata
89 where id_evento = evento_id;
90 end loop;
91 close eventi;
92 end loop;
93 close sessioni;
94 alter table conferenza enable trigger all;
95 alter table sessione enable trigger all;
96 alter table intervento enable trigger all;
97 alter table intervallo enable trigger all;
98 alter table evento enable trigger all;
99 alter table programma enable trigger all;
100 raise notice 'Slittamento completato';
101 exception
102 when others then
103 raise notice '%', sqlerrm;
104 end;
105 $$ 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 $$
12 begin
13 return query
14 select o.id_organizzatore, o.nome, o.cognome, o.email, o.titolo, e.sigla
15 from organizzatore o join ente_conferenza ec natural join ente e
16 on o.id_ente = ec.id_ente
17 where ec.id_conferenza = conferenza
18 GROUP by e.sigla;
19 end;
20 $$ 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 $$
8  declare
9      totale int;
10 begin
11     select count(*) into totale
12     from intervento
13     where date_part('month',inizio) = mese and date_part('year',inizio) = anno;
14
15     return query
16     select e.sigla, (count(*)*100/totale)::text || '%'
17     from intervento i join speaker s
18     on i.id_speaker = s.id_speaker join ente e
19     on s.id_ente = e.id_ente
20     where date_part('month',inizio) = mese and date_part('year',inizio) = anno
21     group by e.sigla;
22 end;
23 $$
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;

```

3.4.6 Sessioni

```

1  create view sessioni as
2  select s.titolo as Sessione, s.inizio, s.fine, c.titolo as Conferenza, s1.nome
   from sessione s, conferenza c, sala s1

```

```
3  where s.id_conferenza=c.id_conferenza and s.id_sala=s1.id_sala  
4  order by s.id_conferenza, s.inizio;
```

Appendice A

Dizionari

A.1 Dizionario dei dati

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

Continua nella prossima pagina

Continua dalla pagina precedente

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

Continua nella prossima pagina

Continua dalla pagina precedente

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

Continua nella prossima pagina

Continua dalla pagina precedente

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

A.2 Dizionario delle associazioni

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

Continua nella pagina successiva

Continua dalla pagina precedente

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

Continua nella pagina successiva

Continua dalla pagina precedente

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

A.3 Dizionario dei vincoli

Vincolo	Tipo	Descrizione
----------------	-------------	--------------------