Documentazione progetto Meeting Planner

Realizzato da:

Francesco Falleroni 261164

Sommario

1. Formalizzazione e analisi dei requisiti

Lo scopo della base di dati, secondo le specifiche descritte dal professore, è quello di poter creare un sistema in grado di gestire e organizzare, tramite le operazioni necessari, l'organizzazione delle sale riunione e dei relativi partecipanti.

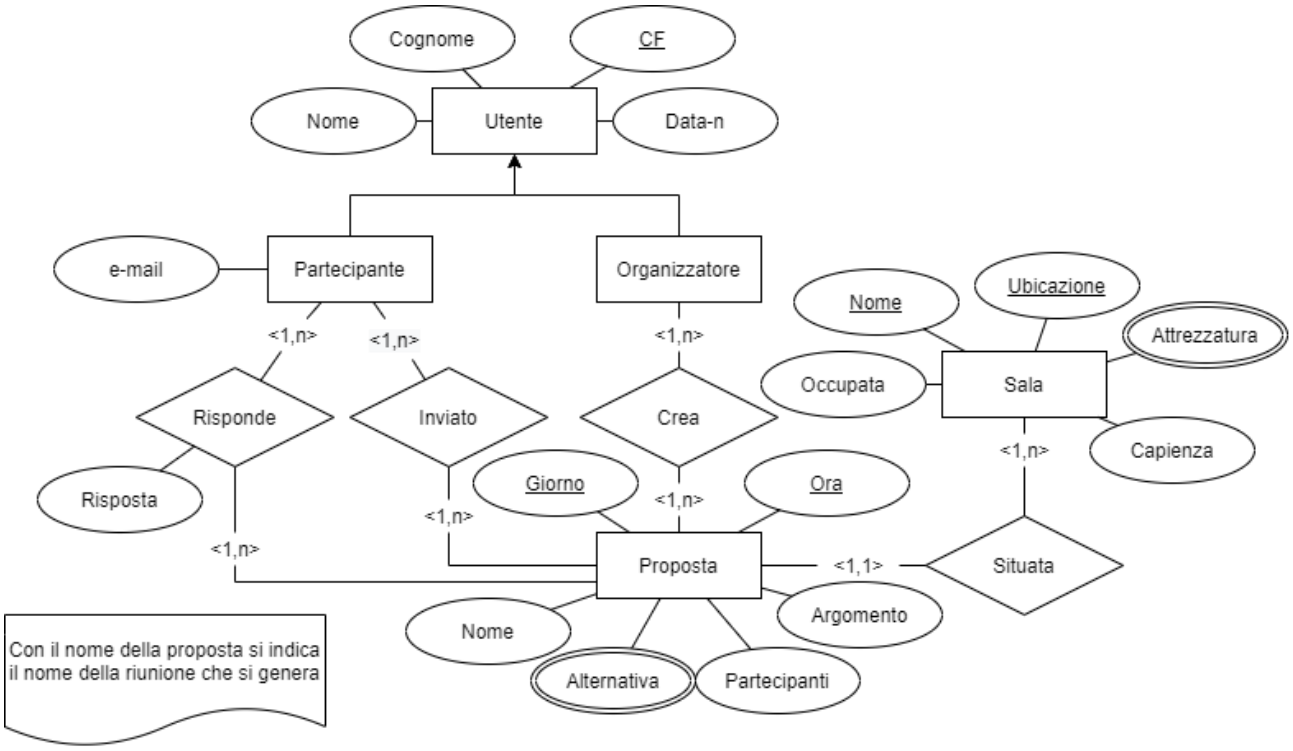
Lo scopo della base di dati è quello di presentare una serie di sale riunioni in cui è possibili vederne la disponibilità, i partecipanti e il momento specifico in cui è occupata per la riunione. In questo contesto si differiscono 2 tipi di utenti:

- Organizzatori, coloro che propongono e organizzano la riunione
- Partecipanti , utenti che sono riconosciuti tramite la loro e-mail

Terminologia	*Descrizione*	*Sinonimo*	*Collegamento*
Organizzatore	Persona fisica che ha lo scopo di creare la proposta di una riunione	Amministratore	Proposta, Riunione
Partecipante	Utente generico che sceglie di partecipare o no a una riunione specifica tramite una proposta ricevuta	Utente	Proposta
Proposta	Invito rilasciato al partecipante in cui sono contenute informazioni relative alla riunione specifica e a una serie di alternative	Invito	Partecipante, Organizzatore
Sala	Luogo in cui si svolgerà la riunione	Luogo	Riunione

2. Progettazione concettuale modello Entità-Relazione

Modello Entità-Relazionale



(Figura 1: schema entità relazione iniziale)

Sopra (Figura 1) è presenta una prima versione del modello entità relazione della nostra base di dati in cui si sono riconosciute 5 entità principali aventi ognuna una PK (primary key) e, eventualmente, di una FK (chiave esterna) :

- Organizzatore → Codice fiscale
- Partecipante → Codice fiscale
- Proposta → Giorno + ora
- Sala → Nome + Ubicazione

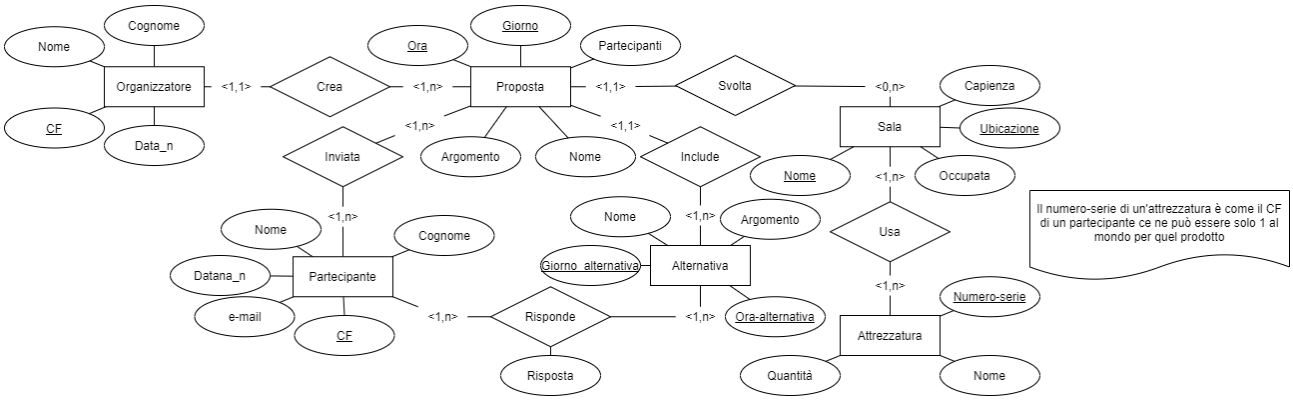
3. Vincoli non esprimibili nel modello ER

Vincoli individuati durante lo svolgimento del progetto:

1. Esiste una sola sala con un nome specifico in un punto specifico dell'edificio dove si svolgerà la riunione
2. Ogni proposta non è identica alle precedenti
3. Per poter far partire una riunione è necessario un numero minimo di partecipanti
4. Ogni alternativa appartenente a una proposta è differente dalle altre

4. Ristrutturazione e ottimizzazione del modello ER

Modello Entità-Relazionale Ricostruito



(Figura 2)

Ricostruzione e eliminazione di attributi multivalore

Si è effettuata una "eliminazione" degli attributi multivalore **attrezzatura** e **alternativa**. Generando così delle nuove entità collegate tramite le relazioni:

- **Include** tra la "Proposta" contiene le l'"Alternativa" ;
- **Usa** tra la "Sala" che possiede l'"Attrezzatura" necessaria;

Successivamente è stata necessaria l'eliminazione della generalizzazione dell'entità di utente, portando il trasferimento dei dati anagrafici in Organizzatore e Partecipante (ricostruzione: Figura 2).

5. Traduzione del modello ER nel modello relazionale

Entità

Organizzatore (ID, nome, cognome, data_n)

Partecipante (ID, nome, cognome, data_n, e-mail)

Proposta (ID, giorno, ora, partecipanti, argomento, nome)

Alternativa (ID, giorno, ora, nome, argomento)

Sala (ID, ubicazione, nome, capienza, occupato)

Attrezzatura (Numero-serie, quantità, nome)

Relazioni

invita (ID-proposta, ID-partecipante)

risponde (ID-alternativa, ID-partecipante, risposta)

usa (ID-sala, NS-attrezzatura)

6. Implementazione tramite SQL

Creazione del DB

```
drop database if exists Meeting2;
create database if not exists Meeting2;
use Meeting2;
```

Creazione della tabella sala

```
create table sala(

      ID integer unsigned primary key auto_increment,
      ubicazione varchar(50),
      nome varchar(50),
      occupato boolean,
      capienza integer not null,
      constraint sala_specifica unique(nome, ubicazione)
);
```

Creazione della tabella delle alternative

```
 create table alternativa(

      ID Integer unsigned primary key auto_increment,
      giorno_alternativa date not null,
      ora_alternativa datetime not null,
      nome varchar(50) not null,
      argomento varchar (50) not null,
      constraint alternativa_specifica unique (ora_alternativa,
      giorno_alternativa)
);
```

Creazione della tabella della proposta

```
create table proposta(
      ID Integer unsigned primary key auto_increment,
      giorno date not null,
      ora datetime not null,
      ID_sala integer unsigned not null,
      ID_alternativa integer unsigned not null,
      argomento varchar(100) not null,
      nome varchar(50) not null,
      partecipanti integer unsigned not null,
      constraint proposta_sala foreign key (ID_sala)
            references sala(ID)
            on update cascade on delete cascade,
      constraint proposta_alternativa foreign key (ID_alternativa)
            references alternativa(ID)
            on update cascade on delete cascade,
      constraint proposta_specifica unique (nome, giorno, ora)
);
```

Creazione della tabella dell'organizzatore

```
create table organizzatore(
      CF CHAR(16) PRIMARY KEY,
      nome varchar(50) not null,
      cognome varchar(50) not null,
      ID_proposta integer unsigned,
      data_n date not null,
```

```
      constraint organizzatore_proposta foreign key (ID_proposta)
            references proposta(ID)
            on update cascade on delete no action
);
```

Creazione della tabella dei partecipanti

```
create table partecipante(
      ID Integer unsigned primary key auto_increment,
      nome varchar(50) not null,
      cognome varchar(50) not null,
      CF varchar(16) not null,
      data_n date not null,
      email varchar(500) unique not null
);
```

Creazione della tabella delle attrezzature

```
create table attrezzatura(
      numero_serie varchar(10) primary key not null,
      quantita integer unsigned,
      nome varchar(50),
      constraint attrezzatra_specifica unique(numero_serie, nome)
);
```

Creazione della tabella usa

```
create table usa(
      ID_sala integer unsigned not null,
      NS_attrezzatura varchar(10) not null,
      PRIMARY KEY (ID_sala, NS_attrezzatura),
      constraint usa_sala foreign key (ID_sala)
            references sala(ID)
            on update cascade on delete cascade,
      constraint usa_attrezzatura foreign key (NS_attrezzatura)
            references attrezzatura(numero_serie)
            on update cascade on delete cascade
);
```

Creazione della tabella invita

```
create table invita(
      ID_proposta integer unsigned not null,
      ID_partecipante integer unsigned not null,
      PRIMARY KEY(ID_proposta, ID_partecipante),
      constraint invtia_proposta foreign key (ID_proposta)
            references proposta(ID)
            on update cascade on delete cascade,
      constraint invita_partecipante foreign key (ID_partecipante)
            references partecipante(ID)
            on update cascade on delete cascade
);
```

Creazione della tabella risponde

```
create table risponde(
      ID_alternativa integer unsigned not null,
      ID_partecipante integer unsigned not null,
      risposta boolean,
      PRIMARY KEY(ID_alternativa, ID_partecipante),
```

```
    constraint risponde_alternativa foreign key (ID_alternativa)
           references alternativa(ID)
           on update cascade on delete no action,
      constraint risponde_partecipante foreign key (ID_partecipante)
           references partecipante(ID)
           on update cascade on delete no action
);
```

7. Implementazione di query, procedure, ecc. …

Query di inserimento

Prima di ogni query è presente una funzione di Delete from <tabella> per svuotare il contenuto delle tabelle.

Ogni quesry è seguita dal risultato di una chiamata per verificare il contenuto di ogni tabella.

```
/*Tabella organizzatori*/
Delete from organizzatore;
Insert into organizzatore (CF, nome, cognome,data_n)
values ("DMRCLD90D16A271O","Claudio", "De Marinis","1990-04-16");
Insert into organizzatore (CF, nome, cognome,data_n)
values ("WSTLLA88C16A271Z","Allan", "West","1988-03-16");
```

Risultato:

	CF	nome	cognome	ID_proposta	data_n
▶	DMRCLD90D16A271O	Claudio	De Marinis	NULL	1990-04-16
	WSTLLA88C16A271Z	Allan	West	NULL	1988-03-16
*	NULL	NULL	NULL	NULL	NULL

```
/*Tabella partecipanti*/
Delete from partecipante;
Insert into partecipante (ID, nome, cognome, CF, data_n,email)
values (1,"Mario", "Rossi","MRARSS89A12H501I", "1989-01-
12","cexaffemi-8711@yopmail.com");
Insert into partecipante (ID, nome, cognome, CF, data_n,email)
values (2,"Luigi", "Veri","VRDLGU93B14F839G","1993-02-14",
"yrreddebem-0865@yopmail.com");
Insert into partecipante (ID, nome, cognome, CF, data_n,email)
values (3,"Fabrizia", "Ferretti","FRRFRZ80M01D810H", "1980-08-
1","nezitissed-2897@yopmail.com");
Insert into partecipante (ID, nome, cognome, CF, data_n,email)
values (4,"Giuseppe", "Galeazzi","GLZGPP00D21A089J", "2000-04-
21","jotteculamu-1513@yopmail.com");
Insert into partecipante (ID, nome, cognome, CF, data_n,email)
values (5,"Maria", "Della Costa","DLLMRA10D52F158K","2010-04-
12","issimmumoco-7729@yopmail.com");
Insert into partecipante (ID, nome, cognome, CF, data_n,email)
values (6,"Maria", "Loggi","LGGMRA02D61A390A", "2002-04-
21","axuffapef-9561@yopmail.com");
```

Risultato:

ID	nome	cognome	CF	data_n	email
1	Mario	Rossi	MRARSS89A12H501I	1989-01-12	cexaffemi-8711@yopmail.com
2	Luigi	Veri	VRDLGU93B14F839G	1993-02-14	yrreddebem-0865@yopmail.com
3	Fabrizia	Ferretti	FRRFRZ80M01D810H	1980-08-01	nezitissed-2897@yopmail.com
4	Giuseppe	Galeazzi	GLZGPP00D21A089J	2000-04-21	jotteculamu-1513@yopmail.com
5	Maria	Della Costa	DLLMRA10D52F158K	2010-04-12	issimmumoco-7729@yopmail.com
6	Maria	Loggi	LGGMRA02D61A390A	2002-04-21	axuffapef-9561@yopmail.com
NULL	NULL	NULL	NULL	NULL	NULL

```
/*Tabella sala*/
Delete from sala;
insert into sala (ID, ubicazione, nome, occupato, capienza)
values (1,"Quarto piano","Sala 3",false, 10);
insert into sala (ID, ubicazione, nome, occupato, capienza)
values (2,"Primo piano","Sala Nord", true, 15);
```

Risultato:

ID	ubicazione	nome	occupato	capienza
1	Quarto piano	Sala 3	0	10
2	Primo piano	Sala Nord	1	15
NULL	NULL	NULL	NULL	NULL

```
/*Tabella attrezzatura*/
Delete from attrezzatura;
insert into attrezzatura (numero_serie, quantita, nome)
values ("LVL-01",1,"Lavagna Lim");
insert into attrezzatura (numero_serie, quantita, nome)
values ("DSA-02",2,"Distrigutori acqua");
insert into attrezzatura (numero_serie, quantita, nome)
values ("SEI-10",10,"Sedie");
insert into attrezzatura (numero_serie, quantita, nome)
values ("TVLR-01",1,"Tavolo per riunioni");
insert into attrezzatura (numero_serie, quantita, nome)
values ("PR-01",1,"Proiettore");
insert into attrezzatura (numero_serie, quantita, nome)
values ("SC-02",2,"Casse audio");
```

Risultato

numero_serie	quantita	nome
DSA-02	2	Distrigutori acqua
LVL-01	1	Lavagna Lim
PR-01	1	Proiettore
SC-02	2	Casse audio
SEI-10	10	Sedie
TVLR-01	1	Tavolo per riunioni
NULL	NULL	NULL

```
/*Tabella Alternativa*/
Delete from alternativa;
insert into alternativa (ID, giorno_alternativa, ora_alternativa, nome, argomento)
values (1, "2020-06-05", '14:00:00',"Controllo annuale", "Controllo delle entrate
e uscite semestrali");
insert into alternativa (ID, giorno_alternativa, ora_alternativa, nome, argomento)
values (2, "2020-02-28", '13:00:00',"Checking antinfurtuni", "Spiegazione norme
per evitare infortuni e urti,");
```

Risultato:

ID	giorno_alternativa	ora_alternativa	nome	argomento
1	2020-06-05	2014-00-00 00:00:00	Controllo annuale	Controllo delle entrate e uscite semestrali
2	2020-02-28	2013-00-00 00:00:00	Checking antinfurtuni	Spiegazione norme per evitare infortuni e urti,
NULL	NULL	NULL	NULL	NULL

```
/*Tabella Proposta*/
Delete from proposta;
insert into proposta (ID, giorno, ora, ID_sala, ID_alternativa, argomento, nome,
partecipanti)
values (1, "2020-01-24",'12:00:00', 1, 2, "Rafforzamento Collaborativo" ,
"Riunione Motivazionale", 20);
insert into proposta (ID, giorno, ora, ID_sala, ID_alternativa, argomento, nome,
partecipanti)
values (2, "2020-12-10",'10:00:00', 2, 2, "Riunione Formativa","Introduzione",
40);
```

Risultato:

ID	giorno	ora	ID_sala	ID_alternativa	argomento	nome	partecipanti
1	2020-01-24	2012-00-00 00:00:00	1	2	Rafforzamento Collaborativo	Riunione Motivazionale	20
2	2020-12-10	2010-00-00 00:00:00	2	2	Riunione Formativa	Introduzione	40
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
/*Tabella usa*/
Delete from usa;
insert into usa (ID_sala, NS_attrezzatura)
values (1,"LVL-01");
insert into usa (ID_sala, NS_attrezzatura)
values (1,"DSA-02");
insert into usa (ID_sala, NS_attrezzatura)
values (1,"SEI-10");
insert into usa (ID_sala, NS_attrezzatura)
values (2,"TVLR-01");
insert into usa (ID_sala, NS_attrezzatura)
values (2,"PR-01");
```

Risultato:

ID_sala	NS_attrezzatura
1	DSA-02
1	LVL-01
2	PR-01
1	SEI-10
2	TVLR-01
NULL	NULL

```
/*Tabella Invita*/
Delete from invita;
insert into invita (ID_proposta, ID_partecipante)
values (1, 1);
insert into invita (ID_proposta, ID_partecipante)
values (1, 2);
insert into invita (ID_proposta, ID_partecipante)
values (1, 3);
insert into invita (ID_proposta, ID_partecipante)
values (2, 4);
insert into invita (ID_proposta, ID_partecipante)
values (2, 5);
insert into invita (ID_proposta, ID_partecipante)
values (2, 6);
```

Risultato:

ID_proposta	ID_partecipante
1	1
1	2
1	3
2	4
2	5
2	6
NULL	NULL

```
/*Tabella Risponde*/
Delete from rispinde;
insert into risponde (ID_alternativa, ID_partecipante, risposta)
values (1, 1, true);
insert into risponde (ID_alternativa, ID_partecipante, risposta)
values (1, 2, true);
insert into risponde (ID_alternativa, ID_partecipante, risposta)
values (1, 3, false);
insert into risponde (ID_alternativa, ID_partecipante, risposta)
values (2, 4, true);
insert into risponde (ID_alternativa, ID_partecipante, risposta)
values (2, 5, false);
insert into risponde (ID_alternativa, ID_partecipante, risposta)
values (2, 6, null);
```

Risultato:

ID_alternativa	ID_partecipante	risposta
1	1	1
1	2	1
1	3	0
2	4	1
2	5	0
2	6	NULL
NULL	NULL	NULL

Query di ricerca

10. Calcolo dell'alternativa di una riunione che ha ricevuto maggiori adesioni

In questa query si usa la funzione **max(partecipanti)** usato nella tabella proposta si estrarrà quella con il maggior numero di adesioni all'interno del DB.

```
-- estrazione della proposta con maggiori adesioni

select p.nome, p.giorno, p.ora , max(partecipanti) as adesioni from proposta
p;
```

Risultato:

nome	giorno	ora	adesioni
Riunione Motivazionale	2020-01-24	2012-00-00 00:00:00	40

11. Individuazione dei partecipanti che non hanno ancora risposto a un invito

Questa query effettua una ricerca tramite il campo risposta della tabella Risponde e riposta tutti i partecipanti con il campo null (vuoto ➔ ancora non hanno risposto).

```
-- ricerca dei partecipanti che non hanno ancora confermato l'invito

select p.nome, p.cognome, p.email from partecipante p
join risponde r on (p.ID = r.ID_partecipante)
where r.risposta is null;
```

Risultato:

nome	cognome	email
Maria	Loggi	axuffapef-9561@yopmail.com

Procedure

1. Inserimento di una sala riunione

L'inserimento si effettua tramite la specifica del piano in cui si svolgerà la sala, il nome della sala e la sua capienza.

```
-- inserimento di una sala riunioni nuova

delimiter $
drop procedure if exists inserimento;
create procedure inserimento(
      in piano varchar(50),
      in nome_sala varchar(50),
      in numero integer unsigned
)

begin
      insert into sala (ubicazione, nome, capienza)
    values (piano, nome_sala, numero);
end$

delimiter ;

call inserimento("terzo piano", "sala grande", 10);

select * from sala;
```

Risultato:

ID	ubicazione	nome	occupato	capienza
1	Quarto piano	Sala 3	0	10
2	Primo piano	Sala Nord	1	15
3	terzo piano	sala grande	NULL	10
NULL	NULL	NULL	NULL	NULL

2. Associazione dell'attrezzatura a una sala riunione

L'associazione avviene tramite un inserimento dell'ID_sala e dell'NS_Attrezzatura all'interno della tabella della relazione che collega attrezzatura e sala.

```
-- associazione di un'attrezzatura
-- a una sala riunioni
delimiter $
drop procedure if exists associa;
create procedure associa(
      in sala integer unsigned,
      in attrezzatura varchar (19)
)

begin
      insert into usa (ID_sala, NS_attrezzatura)
```

```
      values(sala, attrezzatura);
end$

delimiter ;

call associa("2", "LVL-01");

select * from usa;
```

Risultato:

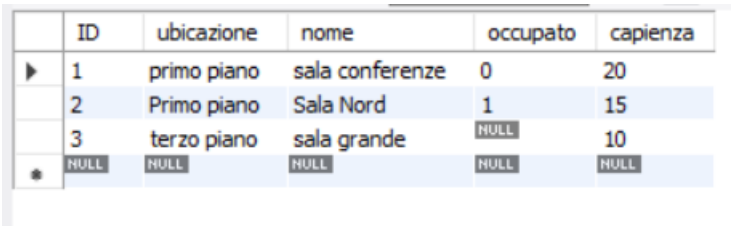
	ID_sala	NS_attrezzatura
▶	1	DSA-02
	1	LVL-01
	2	LVL-01
	2	PR-01
	1	SEI-10
	2	TVLR-01
*	NULL	NULL

3. Creazione (modifica) dell'ubicazione di una sala riunione

Per la modifica dell'ubicazione di una sala si passano le informazioni relative alla nuova ubicazione della riunione (nuova_ubicazione, nuova_sala, nuova_capienza, ID_ricerca) e, tramite la funzione di **update**, si esegue una ricerca della sala con ID_ricerca e poi si aggiorna con le nuova informazioni.

```
-- modifica dell'ubicazione di una sala riunioni

delimiter $
drop procedure if exists modifica;
create procedure modifica(
      in nuova_ubicazione varchar(50),
      in nuova_sala varchar(50),
      in nuova_capienza integer unsigned,
      in ID_ricerca integer unsigned
)

 begin
      update sala
            set ubicazione = nuova_ubicazione, nome = nuova_sala, capienza =
      nuova_capienza
      where ID = ID_ricerca;
 end$

 delimiter ;

 call modifica("primo piano", "sala conferenze", 20, 1);

 select * from sala;
```

Risultato:



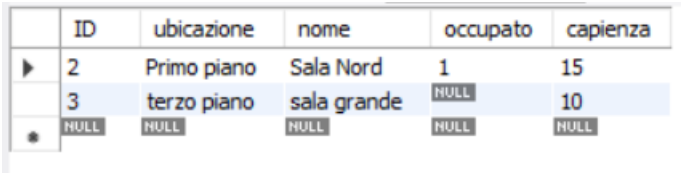
	ID	ubicazione	nome	occupato	capienza
▶	1	primo piano	sala conferenze	0	20
	2	Primo piano	Sala Nord	1	15
	3	terzo piano	sala grande	NULL	10
*	NULL	NULL	NULL	NULL	NULL

4. Cancellazione di una sala riunione

La cancellazione avviene tramite la ricerca dell'ID della sala da cancellare seguita dalla chiamata della comando **delete**.

```
-- cancellazione di una sala riunioni

delimiter $
drop procedure if exists elimina;
create procedure elimina(
      in ID_cancellazione integer unsigned
)

begin
      delete from sala where ID = ID_cancellazione;
end$

delimiter ;

call elimina(1);

select * from sala;
```

Risultato:



	ID	ubicazione	nome	occupato	capienza
▶	2	Primo piano	Sala Nord	1	15
	3	terzo piano	sala grande	NULL	10
*	NULL	NULL	NULL	NULL	NULL

5. Ricerca di una sala riunione libera in un giorno specifico a una data ora e con una capienza minima

In questa procedura si usano i valori data_cercata e ora_cercata per cercare una sala presente nel DB a quel giorno e quell'ora e poi si usa il campo minimo per vedere quali sale sono abbastanza grandi.

```
-- Ricerca di una sala libera
-- in un giorno specifico e con minima capienza
delimiter $
drop procedure if exists cerca;
create procedure cerca(
      in minimo integer unsigned,
      in data_cercata date,
      in ora_cercata datetime
)

begin
      select s.nome, s.ubicazione from sala s
      join proposta p on (p.ID_sala = s.ID)
```

```
      where s.capienza >= minimo and p.giorno = data_cercata and p.ora =
ora_cercata and s.occupato = false;
end$

delimiter ;

call cerca (5,"2020-01-24",'12:00:00');
```

Risultato:

	nome	ubicazione
▶	Sala 3	Quarto piano

6. Creazione di un invito con varie alternative a un giorno e con un orario specifico

In questa caso si usano due procedure:

- La prima crea l'alternativa della proposta di riunione tramite i nuovi valori giorno, ora, nome e argomento
- La seconda invece crea ovviamente la proposta con i valori giorno_proposta, ora_proposta, sala_proposta, alternativa_proposta (qui si passerà l'ID dell'alternativa creata con la procedura precedente), argomento_proposta, nome_proposta e partecipanti_proposta

```
-- creazione di una proposta e alternativa con specifica di giorno e ora

delimiter $
 #creazione alternativa
 drop procedure if exists alternativa;
 create procedure alternativa(
      in giorno date,
      in ora datetime,
      in nome varchar(50),
      in argomento varchar (100)
 )

  begin
      insert into alternativa (giorno_alternativa, ora_alternativa, nome,
argomento)
    values(giorno, ora, nome, argomento);
  end$

#creazione proposta
drop procedure if exists proposta;
 create procedure proposta(
      in giorno_proposta date,
      in ora_proposta datetime,
      in sala_proposta integer unsigned,
      in alternativa_proposta integer unsigned,
      in argomento_proposta varchar(100),
      in nome_proposta varchar(50),
      in partecipanti_proposta integer unsigned
 )

 begin
      insert into proposta (giorno, ora, ID_sala, ID_alternativa, argomento,
      nome, partecipanti)
```

```
      values(giorno_proposta, ora_proposta, sala_proposta,
      alternativa_proposta, argomento_proposta, nome_proposta,
      partecipanti_proposta);
 end$

 delimiter ;

call alternativa("2020-08-15", '08:00:00', "Test", "Test Chiamata Alt");
call proposta("2020-12-11", '10:05:00', 1, 3, "Test", "Test Chiamata Prop",
2);

 select * from alternativa, proposta;
```

Risultato:

ID	giorno_alternativa	ora_alternativa	nome	argomento
1	2020-06-05	2014-00-00 00:00:00	Controllo annuale	Controllo delle entrate e uscite semestrali
2	2020-02-28	2013-00-00 00:00:00	Checking antinfurtuni	Spiegazione norme per evitare infortuni e urti,
3	2020-08-15	2008-00-00 00:00:00	Test	Test Chiamata Alt
NULL	NULL	NULL	NULL	NULL

ID	giorno	ora	ID_sala	ID_alternativa	argomento	nome	partecipanti
1	2020-01-24	2012-00-00 00:00:00	1	2	Rafforzamento Collaborativo	Riunione Motivazionale	20
2	2020-12-10	2010-00-00 00:00:00	2	2	Riunione Formativa	Introduzione	40
3	2020-12-11	2010-05-00 00:00:00	1	3	Test	Test Chiamata Prop	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7.Verifica di un partecipante a una riunione

La verifica delle partecipazioni a una riunione avviene tramite la ricerca di una proposta con il valore riunione e, risalendo le relazioni, si controllano tutti i partecipanti che hanno dato una risposta (può essere si/no).

```
-- verifica delle partecipazioni a una riunione
-- da parte di un utente

delimiter $
drop procedure if exists verifica;
create procedure verifica(
    in riunione  integer unsigned
)

begin

      select p.nome, p.cognome, p.CF from partecipante p
      join risponde r on (p.ID = r.ID_partecipante)
      join alternativa a on (r.ID_alternativa = a.ID)
      join proposta pr on (a.ID = pr.ID_alternativa)
      where pr.ID = riunione and r.risposta is not null;

end$

delimiter ;

call verifica(2);
```

Risultato:

	nome	cognome	CF
▶	Giuseppe	Galeazzi	GLZGPP00D21A089J
	Maria	Della Costa	DLLMRA10D52F158K

8. Registrazione delle partecipazioni in un invito

Per questa procedura si esegue una ricerca del partecipante e di un'alternativa a cui non ha dato la sua partecipazione e tramite p_risposta si esegue una modifica del campo risposta relativa a quell'utente.

```
-- registrazione di un partecipante a un'alterativa

delimiter $
drop procedure if exists conferma;
create procedure conferma(
      in alternativa integer unsigned,
      in ID_p integer unsigned,
      in p_rispsota boolean
)

begin
      update risponde
      set risposta = p_rispsota
      where ID_partecipante = ID_p and ID_alternativa = alternativa;
end$

delimiter ;

call conferma(2,5,true);

select * from risponde;
```

Risultato:

	ID_alternativa	ID_partecipante	risposta
▶	1	1	1
	1	2	1
	1	3	0
	2	4	1
	2	5	1
	2	6	NULL
*	NULL	NULL	NULL

9. Estrazione dei partecipanti a una riunione

Qui si effettua una ricerca, tramite il valore riunione, per sapere di quale proposta di riunione si intende sapere il numero di adesioni che possono essere positive o negative.

```
-- estrazione delle adesioni a
-- ogni alternativa di una riunione

delimiter $
drop procedure if exists estrai_partecipanti;
create procedure estrai_partecipanti(
      in riunione integer unsigned
)

begin
      select p.nome, p.cognome, p.CF from partecipante p
```

```
      join risponde r on (p.ID = r.ID_partecipante)
      join alternativa a on (r.ID_alternativa = a.ID)
      join proposta pr on (a.ID = pr.ID_alternativa)
      where pr.ID = riunione and r.risposta is not null;
end$

delimiter ;

call estrai_partecipanti(1);
```

Risultato:

	nome	cognome	CF
▶	Giuseppe	Galeazzi	GLZGPP00D21A089J
	Maria	Della Costa	DLLMRA10D52F158K

12. Conferma di una riunione e dell'alternativa

Per questa query si decide di impostare il valore risposta a true di un utente specifico a un'alternativa specifica perché tramite l'ID di un partecipante posso vedere a quale proposta parteciperà quell'utente.

```
-- conferma di una proposta e dell'alternativa

delimiter $
drop procedure if exists conferma_pa;
create procedure conferma_pa(
      in partecipante integer unsigned,
      in alternativa integer unsigned,
      in r boolean
)

begin
      update risponde
      set risposta = r
      where ID_alternativa = alternativa and ID_partecipante = partecipante;
end$

delimiter ;

 call conferma_pa(3, 2, true);

 select * from risponde;
```

Risultato:

	ID_alternativa	ID_partecipante	risposta
▶	1	1	1
	1	2	1
	1	3	0
	2	4	1
	2	5	1
	2	6	NULL
*	NULL	NULL	NULL

13. Estrazione dei partecipanti effettivi a una riunione

Si usa lo stesso concetto della procedura numero 9 con la differenza che nel where il caso di ricerca di della risposta è risposta = true.

```
-- estarzione dei partecipanti effettivi a una riunione

delimiter $
drop procedure if exists estrai;
create procedure estrai(
      in ID_estrazione integer unsigned
)

begin
      select p.nome, p.cognome, p.email from partecipante p
      join risponde r on (p.ID = r.ID_partecipante)
      join alternativa a on (r.ID_alternativa = a.ID)
      join proposta pr on (a.ID = pr.ID_alternativa)
      where pr.ID = ID_estrazione and r.risposta = true;
end$

delimiter ;

call estrai(1);
```

Risultato:

	nome	cognome	email
▶	Giuseppe	Galeazzi	jotteculamu-1513@yopmail.com
	Maria	Della Costa	issimmumoco-7729@yopmail.com