



Basi di Dati, A.A. 2015/2016
Corso di Laurea Magistrale in Ingegneria Informatica
Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

Homework 4 – Progettazione Fisica

Data di consegna: 19 maggio 2016

Gruppo M&M	Progetto EasyReg	
Cognomi Maistro	Nomi Maria	Numeri di matricola 123456



Schema Relazionale

In Figura 1 viene presentato lo schema relazionale.

[Nota: Indicare in questa sezione eventuali modifiche dello schema relazionale opportunamente commentate]



Schema fisico

Di seguito sono riportate le istruzioni SQL per l'implementazione della base di dati illustrata dallo schema relazionale in Figura 1.

```
-- Creazione del Database
CREATE DATABASE ecir2016 OWNER ecir2016 ENCODING = 'UTF8';

-- Creo un dominio per l'indirizzo email
CREATE DOMAIN emailaddress AS character varying(254)
    CONSTRAINT properemail CHECK (((VALUE)::text ~* '^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'::text));

-- Creo nuovi tipi di dato
CREATE TYPE actiontype AS ENUM (
    'Payment',
    'RetryPayment',
    'ItalianPayment',
    'FreePayment'
);

CREATE TYPE associationlabels AS ENUM (
    'BcsIrsq',
    'AcmSigir',
    'Clef'
);

CREATE TYPE feetype AS ENUM (
    'Early',
    'Regular',
    'Late',
    'Social',
    'Workshop',
    'Tutorial',
    'Free'
);

CREATE TYPE genderlabels AS ENUM (
    'Male',
    'Female',
    'NotApplicable'
);

CREATE TYPE holder AS ENUM (
    'Organization',
    'Participant'
);

CREATE TYPE jobtitlelabels AS ENUM (
    'Professor',
    'MasterStudent',
    'PhDStudent',
    'PostDocStudent',
    'Researcher',
    'Industry',
    'Other'
);

CREATE TYPE organizationtype AS ENUM (
    'University',
    'Industry'
);

CREATE TYPE prefixlabels AS ENUM (
    'Dr',
    'Miss',
    'Mr',
    'Mrs',
```

```

        'Ms',
        'Prof'
    );

CREATE TYPE specialneedscategory AS ENUM (
    'Dietary',
    'TShirt',
    'Other'
);

CREATE TYPE statustype AS ENUM (
    'Success',
    'Fail',
    'EmailSent',
    'TimeOut',
    'Reimbursed'
);

-- Creo le tabelle
-- Nota: TIMESTAMPTZ è l'abbreviazione di timestamp with time zone
CREATE TABLE Registration(
    Id UUID,
    Checked BOOLEAN NOT NULL,
    Paid BOOLEAN NOT NULL,
    SeparateItalianProcess BOOLEAN NOT NULL,
    TimeStamp TIMESTAMPTZ NOT NULL,
    TotalAmount FLOAT NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE FreeRegistration(
    Code VARCHAR(20),
    Availability BOOLEAN NOT NULL,
    Description TEXT,
    PRIMARY KEY (Code)
);

CREATE TABLE Use(
    Code VARCHAR(20),
    RegistrationId UUID,
    PRIMARY KEY (Code),
    FOREIGN KEY (Code) REFERENCES FreeRegistration(Code),
    FOREIGN KEY (RegistrationId) REFERENCES Registration(Id)
);

CREATE TABLE Log(
    Id BIGSERIAL,
    Action ACTIONTYPE NOT NULL,
    RequestTimestamp TIMESTAMPTZ NOT NULL,
    RequestText TEXT,
    ReplyTimestamp TIMESTAMPTZ,
    ReplyText TEXT,
    Status STATUSTYPE,
    RegistrationId UUID NOT NULL,
    PRIMARY KEY (Id),
    FOREIGN KEY (registrationId) REFERENCES Registration(Id)
);

CREATE TABLE Fee(
    Id VARCHAR(20),
    Type FEETYPE NOT NULL,
    Price FLOAT,
    PRIMARY KEY (Id)
);

CREATE TABLE Participate(
    RegistrationId UUID,
    FeeId VARCHAR(20),
    PRIMARY KEY (RegistrationId, FeeId),
    FOREIGN KEY (RegistrationId) REFERENCES Registration(Id),

```

```

        FOREIGN KEY (FeeId) REFERENCES Fee(Id)
    );

CREATE TABLE Pay(
    RegistrationId UUID,
    FeeId VARCHAR(20),
    Multiplicity SMALLINT NOT NULL,
    PRIMARY KEY (RegistrationId, FeeId),
    FOREIGN KEY (RegistrationId) REFERENCES Registration(Id),
    FOREIGN KEY (FeeId) REFERENCES Fee(Id)
);

-- Nota: Country e CountryofBirth hanno tipo VARCHAR(2) perché viene memorizzato il
codice ISO dello stato
CREATE TABLE Invoice(
    Id BIGSERIAL NOT NULL,
    InvoiceHolder HOLDER NOT NULL,
    Name VARCHAR NOT NULL,
    SecondName VARCHAR,
    Gender GENDERLABELS,
    DateofBirth DATE,
    CityofBirth TEXT,
    CountryofBirth VARCHAR(2),
    Street TEXT NOT NULL,
    City TEXT NOT NULL,
    StateRegion TEXT,
    ZIP TEXT NOT NULL,
    Country VARCHAR(2) NOT NULL,
    FiscalCode TEXT,
    NoVatDeclaring BOOLEAN,
    VatNumber TEXT,
    PublicAdministration BOOLEAN,
    TotalAmount FLOAT NOT NULL,
    TimeStamp TIMESTAMPTZ,
    Released BOOLEAN NOT NULL,
    RegistrationId UUID NOT NULL,
    PRIMARY KEY (Id),
    FOREIGN KEY (RegistrationId) REFERENCES Registration(Id)
);

CREATE TABLE AccompanyingPerson(
    Email EMAILADDRESS,
    Prefix PREFIXLABELS NOT NULL,
    FirstName TEXT NOT NULL,
    MiddleName TEXT,
    LastName TEXT NOT NULL,
    PRIMARY KEY (Email)
);

CREATE TABLE Accompany(
    Email EMAILADDRESS,
    RegistrationId UUID,
    FeeId VARCHAR(20),
    PRIMARY KEY (Email, RegistrationId, FeeId),
    FOREIGN KEY (Email) REFERENCES AccompanyingPerson(Email),
    FOREIGN KEY (RegistrationId) REFERENCES Registration(Id),
    FOREIGN KEY (FeeId) REFERENCES Fee(Id)
);

CREATE TABLE SpecialNeeds(
    Id VARCHAR(20),
    Category SPECIALNEEDSCATEGORY NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE Necessitate(
    Email EMAILADDRESS,
    SpecialNeedsId VARCHAR(20),
    Other TEXT,
    PRIMARY KEY (Email, SpecialNeedsId),

```

```

    FOREIGN KEY (Email) REFERENCES AccompanyingPerson(Email),
    FOREIGN KEY (SpecialNeedsId) REFERENCES SpecialNeeds(Id)
);

CREATE TABLE Paper(
    Id TEXT,
    Title TEXT,
    PRIMARY KEY (Id)
);

CREATE TABLE Participant(
    Email EMAILADDRESS,
    Prefix PREFIXLABELS NOT NULL,
    JobTitle JOBTITLELABELS NOT NULL,
    FirstName TEXT NOT NULL,
    MiddleName TEXT,
    LastName TEXT NOT NULL,
    BadgeName TEXT NOT NULL,
    TwitterAccount TEXT,
    AssociationName ASSOCIATIONLABELS,
    AssociationNumber VARCHAR(30),
    Password VARCHAR NOT NULL,
    Author BOOLEAN NOT NULL,
    Workphone TEXT,
    Cellphone TEXT,
    Fax TEXT,
    TwitterAuthorization BOOLEAN,
    PersonalDataAuthorization BOOLEAN NOT NULL,
    OrganizationName TEXT NOT NULL,
    DepartmentName TEXT,
    OrganizationType ORGANIZATIONTYPE NOT NULL,
    RegistrationId UUID NOT NULL,
    PRIMARY KEY (Email),
    FOREIGN KEY (RegistrationId) REFERENCES Registration(Id)
);

CREATE TABLE Require(
    Email EMAILADDRESS,
    SpecialNeedsId VARCHAR(20),
    Other TEXT,
    PRIMARY KEY (Email, SpecialNeedsId),
    FOREIGN KEY (Email) REFERENCES Participant(Email),
    FOREIGN KEY (SpecialNeedsId) REFERENCES SpecialNeeds(Id)
);

CREATE TABLE Write(
    Email EMAILADDRESS,
    PaperId TEXT,
    PRIMARY KEY (Email, PaperId),
    FOREIGN KEY (Email) REFERENCES Participant(Email),
    FOREIGN KEY (PaperId) REFERENCES Paper(Id)
);

```

Note per l'implementazione dello schema fisico:

1. Le tabelle devono essere create nell'ordine giusto, cioè prima di creare una tabella con un vincolo di chiave esterna bisogna assicurarsi che la tabella a cui fa riferimento la chiave esterna sia già stata creata;
2. Quando si definisce un vincolo di chiave esterna su un campo di una tabella il tipo di dato deve coincidere con il tipo di dato a cui si fa riferimento.

Trigger

Creo un trigger che mi controlla il vincolo: se l'attributo booleano AuthorizationTwitter di Participant ha valore TRUE allora l'attributo TwitterAccount di Participant non può avere valore NULL.

```

CREATE FUNCTION checkParticipantConstraint() RETURNS trigger AS $$

BEGIN

    -- Controllo che l'account di twitter sia diverso dal valore nullo se l'attributo
    -- twitterAuthorization è TRUE
    IF NEW.twitterAccount IS NULL AND NEW.twitterAuthorization IS TRUE THEN
        RAISE EXCEPTION 'TwitterAccount cannot be null';
    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER checkParticipantConstraint BEFORE INSERT OR UPDATE ON participant
FOR EACH ROW EXECUTE PROCEDURE checkParticipantConstraint();

```

Il seguente trigger controlla che prima di inserire una riga nella tabella Write l'attributo Author nella tabella Participant abbia valore TRUE.

```

CREATE FUNCTION checkAuthorConstraint() RETURNS trigger AS $$

DECLARE
    authorVal BOOLEAN;

BEGIN
    -- Controllo che l'attributo Author di Participant abbia valore TRUE
    SELECT Author INTO authorVal FROM Participant WHERE email = NEW.email;

    IF authorVal IS FALSE THEN
        RAISE EXCEPTION 'Participant can not have a submitted paper since he is not
        an author';
    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER checkAuthorConstraint BEFORE INSERT OR UPDATE ON Write
FOR EACH ROW EXECUTE PROCEDURE checkAuthorConstraint();

```

Esempio di popolamento della base di dati

Tutte le tabelle che contengono le informazioni relative ai partecipanti vengono popolate tramite l'utilizzo di un'applicazione web e la compilazione di un modulo da parte dei partecipanti stessi. Di seguito vengono descritte le istruzioni principali per l'inserimento dei dati in queste tabelle.

1. Controllo che il partecipante non si sia già registrato e che l'indirizzo email con cui si registra non sia già presente nel database:

```
SELECT * FROM Participant WHERE Email = 'participant@example.com';
```

Se l'output di questa query è vuoto allora si potrà procedere con l'inserimento dei dati del partecipante nel database, altrimenti verrà visualizzato un messaggio di errore che informerà l'utente che la mail fornita per la registrazione non è valida.

2. Se il partecipante si iscrive con un codice di registrazione gratuita devo controllare che tale codice sia disponibile:

```
SELECT Available FROM FreeRegistration WHERE Code = 'ECIR2016-Ind-162';
```

Se questa query da come output un valore falso allora dovrò visualizzare un messaggio di errore che avviserà il partecipante che il suo codice di registrazione non è valido, altrimenti procederò con l'inserimento dei dati nel database.

3. Genero l'id della registrazione e procedo con l'inserimento dei dati nella tabella Registration:

```
INSERT INTO registration (id, checked, paid, separateitalianprocess, timestamp,
totalamount) VALUES ('04a0fbba-4ddc-4512-ab1b-9c433a36be14', FALSE, FALSE, FALSE,
NOW(), 480);
```

4. Procedo con l'inserimento dei dati riguardanti la partecipazione agli eventi della conferenza, per ogni evento dovrò aggiungere una tupla nella tabella Partecipate:

```
INSERT INTO participate (RegistrationId, FeeId) VALUES ('04a0fbba-4ddc-4512-ab1b-
9c433a36be14', 'Banquet');
INSERT INTO participate (RegistrationId, FeeId) VALUES ('04a0fbba-4ddc-4512-ab1b-
9c433a36be14', 'BIRW1');
INSERT INTO participate (RegistrationId, FeeId) VALUES ('04a0fbba-4ddc-4512-ab1b-
9c433a36be14', 'LiLaT3');
```

5. Popolo la tabella Pay con la tipologia di iscrizione:

```
INSERT INTO pay (RegistrationId, FeeId, Multiplicity) VALUES ('04a0fbba-4ddc-4512-
ab1b-9c433a36be14', 'AssociationE', 1);
```

se il partecipante si iscrive con la tipologia di iscrizione gratuita dovrò inserire i valori corrispondenti nella tabella Use e aggiornare la tabella FreeRegistration:

```
INSERT INTO Use (Code, RegistrationId) VALUES ('ECIR2016-Ind-162', '04a0fbba-4ddc-
4512-ab1b-9c433a36be14');
UPDATE FreeRegistration SET Available = FALSE WHERE Code = 'ECIR2016-Ind-162';
```

6. Inserisco le informazioni riguardanti la fattura:

```
INSERT INTO Invoice (id, InvoiceHolder, Name, SecondName, Gender, DateofBirth,
CityofBirth, CountryofBirth, Street, City, StateRegion, ZIP, Country, FiscalCode,
NoVatDeclaring, VatNumber, PublicAdministration, Timestamp, TotalAmount, Released,
RegistrationId) VALUES (DEFAULT, 'Participant', 'Name', 'Surname', 'Male', '1988-
03-03', 'Padova', 'IT', 'Via Gradenigo 6 B', 'Padova', NULL, '35131', 'IT', NULL,
NULL, NULL, NULL, NOW(), 480, FALSE, '04a0fbba-4ddc-4512-ab1b-9c433a36be14');
```

Si noti che per ottenere in ritorno il valore inserito di default, in questo caso id, si può usare l'istruzione RETURNING:

```
INSERT INTO Invoice (id, InvoiceHolder, Name, SecondName, Gender, DateofBirth,
CityofBirth, CountryofBirth, Street, City, StateRegion, ZIP, Country, FiscalCode,
NoVatDeclaring, VatNumber, PublicAdministration, Timestamp, TotalAmount, Released,
RegistrationId) VALUES (DEFAULT, 'Participant', 'Name', 'Surname', 'Male', '1988-
03-03', 'Padova', 'IT', 'Via Gradenigo 6 B', 'Padova', NULL, '35131', 'IT', NULL,
NULL, NULL, NULL, NOW(), 420, FALSE, '04a0fbba-4ddc-4512-ab1b-9c433a36be14')
RETURNING id;
```

7. Inserisco le informazioni dell'accompagnatore (se presente), gli eventi a cui partecipa, le sue esigenze speciali e le quote di partecipazione che dovranno essere aggiunte all'iscrizione:

```
INSERT INTO AccompanyingPerson (Email, Prefix, FirstName, MiddleName, LastName)
VALUES ('accperson@example.com', 'Ms', 'FirstName', NULL, 'LastName');
INSERT INTO Accompany (Email, RegistrationId, FeeId) VALUES
('accperson@example.com', '04a0fbba-4ddc-4512-ab1b-9c433a36be14', 'Banquet');
INSERT INTO Necessitate (Email, SpecialNeedsId, Other) VALUES
('accperson@example.com', 'Vegetarian', 'Vegetarian and not vegan diet');
INSERT INTO pay (RegistrationId, FeeId, Multiplicity) VALUES ('04a0fbba-4ddc-4512-
ab1b-9c433a36be14', 'Banquet', 1);
```

8. Popolo le tabelle con le informazioni anagrafiche del partecipante e le sue esigenze speciali:

```
INSERT INTO participant (email, prefix, jobtitle, firstname, middlename, lastname,
badgename, twitteraccount, associationname, associationnumber, password, author,
workphone, cellphone, fax, twitterauthorization, personaldataauthorization,
organizationname, departmentname, organizationtype, registrationid) VALUES
('participant@example.com', 'Mr', 'PhDStudent', 'Name', 'MiddelName', 'Lastname',
'Name Lastname', '@example', NULL, NULL, '1478af53849fcce57e86c72e5bcc1601', TRUE,
'54791347614', NULL, NULL, TRUE, TRUE, 'University of Padua', 'Department of
Information Engineering', 'University', '04a0fbba-4ddc-4512-ab1b-9c433a36be14');
INSERT INTO Require (Email, SpecialNeedsId, Other) VALUES
('participant@example.com', 'OtherSpecial', 'I would need a letter for visa
application.');
```



```
INSERT INTO Require (Email, SpecialNeedsId, Other) VALUES
('participant@example.com', 'Vegetarian', NULL);
```

9. Se il partecipante è autore di uno o più articoli inserisco la relativa associazione nella tabella write:

```
INSERT INTO write (email, paperid) VALUES ('participant@example.com', '28');
```

Le altre tabelle (ad eccezione della tabella FreeRegistration) devono essere popolate prima dell'apertura delle registrazioni, infatti contengono informazioni che non riguardano il partecipante e che sono definite dagli organizzatori della conferenza. Inoltre, le informazioni memorizzate in tali tabelle sono necessarie per la compilazione delle tabelle relative ai partecipanti. Ad esempio, per popolare la tabella Write, che contiene l'associazione tra partecipante e paper, è necessario che la tabella Paper contenga già tutti gli identificati degli articoli accettati alla conferenza.

- Tabella delle esigenze speciali:

```
INSERT INTO SpecialNeeds VALUES ('Vegetarian', 'Dietary');
INSERT INTO SpecialNeeds VALUES ('Celiac', 'Dietary');
INSERT INTO SpecialNeeds VALUES ('Vegan', 'Dietary');
INSERT INTO SpecialNeeds VALUES ('Other', 'Dietary');
INSERT INTO SpecialNeeds VALUES ('OtherSpecial', 'Other');
```

- Tabella delle quote di iscrizioni e degli eventi

```
INSERT INTO Fee VALUES ('StudentE', 'Early', '260');
INSERT INTO Fee VALUES ('StudentR', 'Regular', '310');
INSERT INTO Fee VALUES ('StudentL', 'Late', '350');
INSERT INTO Fee VALUES ('AssociationE', 'Early', '420');
INSERT INTO Fee VALUES ('AssociationR', 'Regular', '470');
INSERT INTO Fee VALUES ('AssociationL', 'Late', '490');
INSERT INTO Fee VALUES ('RegularE', 'Early', '470');
INSERT INTO Fee VALUES ('RegularR', 'Regular', '520');
INSERT INTO Fee VALUES ('RegularL', 'Late', '550');
INSERT INTO Fee VALUES ('WorkTutOnlyE', 'Early', '200');
INSERT INTO Fee VALUES ('WorkTutOnlyR', 'Regular', '230');
INSERT INTO Fee VALUES ('WorkTutOnlyL', 'Late', '260');
INSERT INTO Fee VALUES ('MainConfOnlyE', 'Early', '300');
INSERT INTO Fee VALUES ('MainConfOnlyR', 'Regular', '320');
INSERT INTO Fee VALUES ('MainConfOnlyL', 'Late', '340');
INSERT INTO Fee VALUES ('IndDayOnlyE', 'Early', '120');
INSERT INTO Fee VALUES ('IndDayOnlyR', 'Regular', '150');
INSERT INTO Fee VALUES ('IndDayOnlyL', 'Late', '180');
INSERT INTO Fee VALUES ('FreeRegistration', 'Free', '0');
INSERT INTO Fee VALUES ('Banquet', 'Social', '60');
INSERT INTO Fee VALUES ('PosterRec', 'Social', '15');
INSERT INTO Fee VALUES ('EveningRec', 'Social', '10');

INSERT INTO Fee VALUES ('BIRW1', 'Workshop', NULL);
INSERT INTO Fee VALUES ('MultiLingMineW2', 'Workshop', NULL);
INSERT INTO Fee VALUES ('ProActIRW3', 'Workshop', NULL);
INSERT INTO Fee VALUES ('NewsIRW4', 'Workshop', NULL);

INSERT INTO Fee VALUES ('CollaborativeIRT1', 'Tutorial', NULL);
INSERT INTO Fee VALUES ('RecSystemsT2', 'Tutorial', NULL);
INSERT INTO Fee VALUES ('LiLaT3', 'Tutorial', NULL);
INSERT INTO Fee VALUES ('RTBMAT4', 'Tutorial', NULL);
```

- Tabella degli articoli:

```
INSERT INTO Paper VALUES ('2', 'First International Workshop on Recent Trends in
News Information Retrieval (NewsIR'16)');
INSERT INTO Paper VALUES ('3', 'Collaborative Information Retrieval: Concepts,
Models and Evaluation');
INSERT INTO Paper VALUES ('4', 'Group Recommender Systems: State of the Art,
Emerging Aspects and Techniques, and Research Challenges');
INSERT INTO Paper VALUES ('5', 'Real-Time Bidding based Display Advertising:
Mechanisms and Algorithms');
```

```

INSERT INTO Paper VALUES ('6', 'Modeling, Learning and Mining for
Cross/Multilinguality');
INSERT INTO Paper VALUES ('7', 'Bibliometric-enhanced Information Retrieval: 3rd
International BIR Workshop');
INSERT INTO Paper VALUES ('8', 'Proactive Information Retrieval: Anticipating
users'' information need');
INSERT INTO Paper VALUES ('9', 'Living Labs for Online Evaluation: From Theory to
Practice');
INSERT INTO Paper VALUES ('14', 'Multi-task Representation Learning for
Demographic Prediction');
INSERT INTO Paper VALUES ('18', 'Implicit Look-alike Modelling in Display Ads:
Transfer Collaborative Filtering to CTR Estimation');
INSERT INTO Paper VALUES ('20', 'Deep Learning over Categorical Data: A Case Study
on User Response Prediction');
INSERT INTO Paper VALUES ('25', 'A Business Zone Recommender System Based on
Facebook and Urban Planning Data');
INSERT INTO Paper VALUES ('28', 'Does Selective Search Benefit from WAND
Optimization?');

```

- Tabella delle registrazioni gratuite:

```

INSERT INTO FreeRegistration (code, available, description) VALUES ('ECIR2016-KSJ-
497', TRUE, 'Name Latsname <participant@example.com>');

```

Interrogazioni principali

Seleziono le informazioni anagrafiche dei partecipanti necessarie per creare la fattura:

```

SELECT Email, Prefix, FirstName, MiddleName, LastName, InvoiceHolder, Name, SecondName,
Gender, DateofBirth, CityofBirth, CountryofBirth, Street, City, StateRegion, ZIP,
Country, FiscalCode, VatNumber, I.TotalAmount, Released, R.Id, Paid,
SeparateItalianProcess FROM Invoice AS I INNER JOIN Registration AS R ON I.registrationid
= R.id INNER JOIN Participant AS P ON P.registrationid = R.id ORDER BY paid DESC,
Released ASC;

```

Seleziono i nomi dei partecipanti che hanno effettuato il pagamento, la data e l'id del pagamento:

```

SELECT Email, Prefix, FirstName, LastName, ReplyTimestamp::date, Id FROM Log AS L INNER
JOIN Participant AS P ON L.registrationId = P.registrationId WHERE Status = 'Success';

```

Seleziono il numero di partecipanti iscritti per ogni tipologia di quota di iscrizione:

```

SELECT feeId AS feeId, COUNT(*) AS numberofFee FROM Registration AS R INNER JOIN Pay AS P
ON R.id = P.registrationId WHERE paid = TRUE GROUP BY feeId;

```

Output pane		
	Data Output	Explain Messages History
	feeId character varying(20)	numberofFee bigint
1	IndDayOnlyR	2
2	RegularR	15
3	PosterRec	1
4	OneDayOnlyL	1
5	IndDayOnlyL	1
6	StudentE	48
7	AssociationE	21
8	AssociationR	8
9	StudentR	5
10	RegularE	54

Visualizzo l'ammontare relativo alle entrate giornaliere:

```

SELECT replytimestamp::date AS paymentdate, COUNT(*) AS numberofpayments,
SUM(totalAmount) AS totalamount FROM log AS L INNER JOIN registration AS R ON
L.registrationid = R.id WHERE status = 'Success' GROUP BY replytimestamp::date ORDER BY
paymentdate ASC;

```

Output pane			
Data Output	Explain	Messages	History
	paymentdate date	numberofpayments bigint	totalamount double precision
1	2016-01-08	15	4850
2	2016-01-09	2	680
3	2016-01-11	12	3690
4	2016-01-12	14	4740
5	2016-01-13	3	860
6	2016-01-14	6	2370
7	2016-01-15	20	7490
8	2016-01-16	7	2420
9	2016-01-17	1	480
10	2016-01-20	1	480

Visualizzo gli identificatori e i titoli degli articoli che non hanno ancora un autore registrato alla conferenza:

```
SELECT id::int AS id, title FROM paper AS P EXCEPT SELECT paperid::int AS id, title FROM
write AS W INNER JOIN participant AS PR ON W.email = PR.email INNER JOIN Registration AS
R ON PR.registrationId = R.id INNER JOIN paper AS PA ON W.paperid = PA.id WHERE paid =
TRUE ORDER BY id;
```

Output pane		
Data Output	Explain	Messages
	id integer	title text
1	128	Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge
2	191	Supporting Human Answers for Advice-Seeking Questions in CQA Sites

Visualizzo il numero di partecipanti iscritti ad ogni evento:

```
SELECT feeid, COUNT(*) AS participantNumber FROM participate AS P INNER JOIN Log AS L ON
P.registrationId = L.registrationId WHERE L.status = 'Success' GROUP BY feeid;
```

Output pane		
Data Output	Explain	Messages
	feeid character varying(20)	participantnumber bigint
1	RecSystemsT2	38
2	PosterRec	170
3	MultilingMineW2	38
4	BIRW1	37
5	EveningRec	168
6	RTBMAT4	27
7	ProActIRW3	51
8	Banquet	165
9	LiLaT3	48
10	CollaborativeIRT1	54
11	NewsIRW4	69

Visualizzo il numero degli accompagnatori iscritti agli eventi sociali:

```
SELECT feeid, COUNT(*) AS participantNumber FROM accompany AS A INNER JOIN Log AS L ON
A.registrationId = L.registrationId WHERE Status = 'Success' GROUP BY feeid;
```

Output pane			
Data Output		Explain	Messages History
	feeid character varying(20)	participantnumber bigint	
1	EveningRec	2	
2	Banquet	4	
3	PosterRec	1	

Conto il numero di partecipanti che si sono iscritti a tutti i tutorial e a tutti i workshop:

```
SELECT COUNT(*) AS busyparticipant, type FROM (SELECT registrationId, Type FROM
participate AS P INNER JOIN fee AS F ON P.feeid = F.id GROUP BY registrationId, type
HAVING COUNT(*) > 3) AS tmp GROUP BY type;
```

Output pane			
Data Output		Explain	Messages History
	busyparticipant bigint	type feetype	
1	11	Workshop	
2	14	Tutorial	

Visualizzo le esigenze speciali dei partecipanti:

```
SELECT P.email, firstname, lastname, specialneedsid, other FROM require AS R INNER JOIN
participant AS P ON R.email = P.email WHERE specialneedsid = 'OtherSpecial' OR
specialneedsid = 'Other' ORDER BY lastname, firstname;
```

	specialneedsid character varying(20)	other text
	Other	No pepper, please
dran	Other	I am allergic to crustacean based seafood.
	Other	Vegan
	OtherSpecial	NO
	Other	Kosher (full vegetarian will be good too)
	Other	Egg Tomato Cucumber Avocado
	Other	No porc
	Other	Halal Food

Visualizzo i partecipanti che si sono iscritti con registrazione gratuita:

```
SELECT U.code, description, prefix, firstname, middlename, lastname, email,
organizationname, departmentname FROM FreeRegistration AS F INNER JOIN Use AS U ON
F.code = U.code INNER JOIN Participant AS P ON U.registrationid = P.registrationid;
```

Stored Procedure

Funzione che data la mail di un partecipante cancella dal database tutte le informazioni relative a quel partecipante.

```
CREATE FUNCTION deleteparticipant(myemail character varying) RETURNS void
    LANGUAGE plpgsql
    AS $$
DECLARE
    myRegistrationId uuid;
    myAccEmail varchar;
BEGIN
```

```

SELECT RegistrationId INTO myRegistrationId FROM Participant WHERE email =
myEmail;

IF NOT FOUND THEN
    RAISE EXCEPTION 'Participant with email % not found', myEmail;
END IF;

-- Delete accompanying person information
FOR myAccEmail IN
    SELECT email FROM Accompany WHERE RegistrationId = myRegistrationId
LOOP
    DELETE FROM Necessitate WHERE email = myAccEmail;
    DELETE FROM Accompany WHERE email = myAccEmail;
    DELETE FROM AccompanyingPerson WHERE email = myAccEmail;
END LOOP;

DELETE FROM Write WHERE email = myEmail;
DELETE FROM Require WHERE email = myEmail;
DELETE FROM Participant WHERE registrationId = myRegistrationId;
DELETE FROM Invoice WHERE registrationId = myRegistrationId;
DELETE FROM Log WHERE registrationId = myRegistrationId;
DELETE FROM Participate WHERE registrationId = myRegistrationId;
DELETE FROM Pay WHERE registrationId = myRegistrationId;
DELETE FROM Use WHERE registrationId = myRegistrationId;
DELETE FROM Registration WHERE Id = myRegistrationId;

RETURN;

END;
$$;

```

Implementazione JDBC delle interrogazioni principali e visualizzazione

Classe Java che legge e stampa sullo schermo le esigenze speciali dei partecipanti:

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * Lists all the authors in the database
 *
 * @author Maria Maistro
 * @version 1.00
 */
public class readSpecialNeeds {

    /**
     * The JDBC driver to be used
     */
    private static final String DRIVER = "org.postgresql.Driver";

    /**
     * The URL of the database to be accessed
     */
    private static final String DATABASE = "jdbc:postgresql://ims-
pgsql.dei.unipd.it/ecir2016";

    /**
     * The username for accessing the database
     */
    private static final String USER = "";

    /**
     * The password for accessing the database

```

```

    */
    private static final String PASSWORD = "";

    /**
     * The SQL statement to be executed
     */
    private static final String SQL = "SELECT P.email, firstname, lastname,
specialneedsid, other FROM require AS R INNER JOIN participant AS P ON R.email = P.email
WHERE specialneedsid = 'OtherSpecial' OR specialneedsid = 'Other' ORDER BY lastname,
firstname;";

    /**
     * List the special need of the participants
     *
     * @param args
     *         command-line arguments (not used).
     */
    public static void main(String[] args) {

        // the connection to the DBMS
        Connection con = null;

        // the statement to be executed
        Statement stmt = null;

        // the results of the statement execution
        ResultSet rs = null;

        // start time of a statement
        long start;

        // end time of a statement
        long end;

        // "data structures" for the data to be read from the database

        // the data about the participant
        String email = null;
        String firstName = null;
        String lastName = null;
        String specialNeedsId = null;
        String otherText = null;

        try {
            // register the JDBC driver
            Class.forName(DRIVER);

            System.out.printf("Driver %s successfully registered.\n", DRIVER);
        } catch (ClassNotFoundException e) {
            System.out.printf(
                "Driver %s not found: %s.\n", DRIVER, e.getMessage());

            // terminate with a generic error code
            System.exit(-1);
        }

        try {

            // connect to the database
            start = System.currentTimeMillis();

            con = DriverManager.getConnection(DATABASE, USER, PASSWORD);

            end = System.currentTimeMillis();

            System.out.printf(

```

```

        "Connection to database %s successfully established in
        %,d milliseconds.%n",
        DATABASE, end-start);

// create the statement to execute the query
start = System.currentTimeMillis();

stmt = con.createStatement();

end = System.currentTimeMillis();

System.out.printf(
    "Statement successfully created in %,d milliseconds.%n",
    end-start);

// execute the query
start = System.currentTimeMillis();

rs = stmt.executeQuery(SQL);

end = System.currentTimeMillis();

System.out
    .printf("Query %s successfully executed %,d
    milliseconds.%n",
    SQL, end - start);

System.out
    .printf("Query results:%n");

// cycle on the query results and print them
while (rs.next()) {

    // read the email of a participant
    email = rs.getString("email");

    // read the first name of a participant
    firstName = rs.getString("firstname");

    // read the last name of a participant
    lastName = rs.getString("lastname");

    // read the special need id
    specialNeedsId = rs.getString("specialneedsid");

    // read the text other
    otherText = rs.getString("other");

    System.out.printf("- %s, %s, %s, %s, %s%n",
        email, firstName, lastName, specialNeedsId,
        otherText);

}
} catch (SQLException e) {
    System.out.printf("Database access error:%n");

    // cycle in the exception chain
    while (e != null) {
        System.out.printf("- Message: %s%n", e.getMessage());
        System.out.printf("- SQL status code: %s%n", e.getSQLState());
        System.out.printf("- SQL error code: %s%n", e.getErrorCode());
        System.out.printf("%n");
        e = e.getNextException();
    }
} finally {
    try {

        // close the used resources
        if (rs != null) {

```

```

        start = System.currentTimeMillis();

        rs.close();

        end = System.currentTimeMillis();

        System.out
        .printf("Result set successfully closed in %,d
                milliseconds.%n",
                end-start);
    }

    if (stmt != null) {

        start = System.currentTimeMillis();

        stmt.close();

        end = System.currentTimeMillis();

        System.out
        .printf("Statement successfully closed in %,d
                milliseconds.%n",
                end-start);
    }

    if (con != null) {

        start = System.currentTimeMillis();

        con.close();

        end = System.currentTimeMillis();

        System.out
        .printf("Connection successfully closed in %,d
                milliseconds.%n",
                end-start);
    }

    System.out.printf("Resources successfully released.%n");

} catch (SQLException e) {
    System.out.printf("Error while releasing resources.%n");

    // cycle in the exception chain
    while (e != null) {
        System.out.printf("- Message: %s%n", e.getMessage());
        System.out.printf("- SQL status code: %s%n",
            e.getSQLState());
        System.out.printf("- SQL error code: %s%n",
            e.getErrorCode());
        System.out.printf("%n");
        e = e.getNextException();
    }

} finally {

    // release resources to the garbage collector
    rs = null;
    stmt = null;
    con = null;

    System.out.printf("Resources released to the garbage
        collector.%n");
}

System.out.printf("Program end.%n");

```



```
    }  
}
```

Istruzioni da riga di comando per compilare ed eseguire il codice:

```
javac readSpecialNeeds.java  
java -cp .:postgresql-9.4-1201.jdbc4.jar readSpecialNeeds
```