**[CREAZIONE TABLE]**

CREATE DATABASE progettoSSD;

USE progettoSSD;

CREATE TABLE Utente (

ID int(10) NOT NULL AUTO\_INCREMENT,

Nome varchar(255),

Cognome varchar(255),

PRIMARY KEY (ID)

);

CREATE TABLE Prodotto (

ID int(10) NOT NULL AUTO\_INCREMENT,

ListaDesideriID int(10) NOT NULL,

Nome varchar(255),

PRIMARY KEY (ID)

);

CREATE TABLE ListaDesideri (

ID int(10) NOT NULL AUTO\_INCREMENT,

UtenteID int(10) NOT NULL,

Nome varchar(255),

PRIMARY KEY (ID)

);

CREATE TABLE Gruppo (

ID int(10) NOT NULL AUTO\_INCREMENT,

Nome varchar(255),

PRIMARY KEY (ID)

);

CREATE TABLE Partecipazione (

ID int(10) NOT NULL AUTO\_INCREMENT,

UtenteProprietario tinyint(1) NOT NULL,

UtenteID int(10) NOT NULL,

GruppoID int(10) NOT NULL,

PRIMARY KEY (ID)

);

CREATE TABLE Gruppo\_IDListeCondivise (

GruppoID int(10) NOT NULL,

GruppoIndex int(10) NOT NULL,

IDListeCondivise int(10) NOT NULL,

PRIMARY KEY (GruppoID, GruppoIndex)

);

ALTER TABLE Prodotto ADD CONSTRAINT FKProdotto471876 FOREIGN KEY (ListaDesideriID) REFERENCES ListaDesideri (ID);

ALTER TABLE ListaDesideri ADD CONSTRAINT FKListaDesid479326 FOREIGN KEY (UtenteID) REFERENCES Utente (ID);

ALTER TABLE Partecipazione

ADD CONSTRAINT FKPartecipaz792957 FOREIGN KEY (UtenteID) REFERENCES Utente (ID);

ALTER TABLE Partecipazione ADD CONSTRAINT FKPartecipaz714368 FOREIGN KEY (GruppoID) REFERENCES Gruppo (ID);

ALTER TABLE Gruppo\_IDListeCondivise ADD CONSTRAINT FKGruppo\_IDL170216 FOREIGN KEY (GruppoID) REFERENCES Gruppo (ID);

**SOSTITUZIONE RAPIDA**

CREATE database progettoSSD;

USE progettoSSD;

CREATE TABLE Utente (ID int(10) NOT NULL AUTO\_INCREMENT, Nome varchar(255), Cognome varchar(255), PRIMARY KEY (ID));

CREATE TABLE Prodotto (ID int(10) NOT NULL AUTO\_INCREMENT, ListaDesideriID int(10) NOT NULL, Nome varchar(255), PRIMARY KEY (ID));

CREATE TABLE ListaDesideri (ID int(10) NOT NULL AUTO\_INCREMENT, UtenteID int(10) NOT NULL, Nome varchar(255), PRIMARY KEY (ID));

CREATE TABLE Gruppo (ID int(10) NOT NULL AUTO\_INCREMENT, Nome varchar(255), PRIMARY KEY (ID));

CREATE TABLE Partecipazione (ID int(10) NOT NULL AUTO\_INCREMENT, UtenteProprietario tinyint(1) NOT NULL, UtenteID int(10) NOT NULL, GruppoID int(10) NOT NULL, PRIMARY KEY (ID));

CREATE TABLE Gruppo\_ListaDesideri (GruppoID int(10) NOT NULL, ListaDesideriID int(10) NOT NULL, PRIMARY KEY (GruppoID, ListaDesideriID));

ALTER TABLE Prodotto ADD CONSTRAINT FKProdotto471876 FOREIGN KEY (ListaDesideriID) REFERENCES ListaDesideri (ID);

ALTER TABLE ListaDesideri ADD CONSTRAINT FKListaDesid479326 FOREIGN KEY (UtenteID) REFERENCES Utente (ID);

ALTER TABLE Partecipazione ADD CONSTRAINT FKPartecipaz792957 FOREIGN KEY (UtenteID) REFERENCES Utente (ID);

ALTER TABLE Partecipazione ADD CONSTRAINT FKPartecipaz714368 FOREIGN KEY (GruppoID) REFERENCES Gruppo (ID);

ALTER TABLE Gruppo\_ListaDesideri ADD CONSTRAINT FKGruppo\_Lis633240 FOREIGN KEY (GruppoID) REFERENCES Gruppo (ID);

ALTER TABLE Gruppo\_ListaDesideri ADD CONSTRAINT FKGruppo\_Lis683055 FOREIGN KEY (ListaDesideriID) REFERENCES ListaDesideri (ID);

**### DASHBOARD ###**

**---- RECUPERA ID E NOME LISTE PERSONALI (@ GetPersonalLists)----**

SELECT ID, Nome FROM listadesideri AS lista WHERE lista.UtenteID = "1";

**---- RECUPERA ID E NOME GRUPPI PERSONALI (@ GetPersonalGroups) ----**

SELECT ID, Nome FROM gruppo AS g WHERE EXISTS (SELECT UtenteID, GruppoID FROM partecipazione AS p WHERE g.ID = p.GruppoID AND p.UtenteID = '2');

---- @NewList ----

INSERT INTO listadesideri (UtenteID, Nome) VALUES ('1', 'La Mia Nuova Lista');

---- @NewGroup ---- DA RIFARE CON TRIGGER

INSERT INTO gruppo (Nome) VALUES ('Nuovo Gruppo');

SELECT last\_insert\_id() AS id\_nuovo\_gruppo;

* INSERT INTO partecipazione (UtenteProprietario, UtenteID, GruppoID) VALUES ('1','2',id\_nuovo\_gruppo);

CREATE TRIGGER before\_insert\_partecipazione

BEFORE INSERT

ON partecipazione FOR EACH ROW

INSERT INTO gruppo(Nome) VALUES("Nuovo gruppo");

INSERT INTO partecipazione (UtenteProprietario, UtenteID, GruppoID) VALUES ('1','2',id\_nuovo\_gruppo);

* **CON LA STORED PROCEDURE (ma si poteva fare pure con il trigger)**

DELIMITER $$

CREATE PROCEDURE inserimento\_gruppo(IN nome\_gruppo VARCHAR(255), IN id\_user int)

BEGIN

DECLARE id\_gruppo int;

INSERT INTO gruppo (Nome) VALUES (nome\_gruppo);

SET id\_gruppo = last\_insert\_id();

INSERT INTO partecipazione (UtenteProprietario, UtenteID, GruppoID) VALUES (1,id\_user,id\_gruppo);

END $$

call progettossd.inserimento\_gruppo('Gruppo Prova', 3);

**### LISTA ###**

**---- SELEZIONA I PRODOTTI DI UNA LISTA DEI DESIDERI (@ GetList) ----**

SELECT \* FROM prodotto AS pr WHERE EXISTS(SELECT \* FROM listadesideri AS lista WHERE pr.ListaDesideriID = lista.ID AND pr.ListaDesideriID = '1');

* **SELEZIONA NOME LISTA**
* SELECT Nome FROM listadesideri AS lista WHERE lista.ID = '3';

^^^ Così stiamo facendo due query. Se ne può fare anche una sola.

**---- AGGIUNGI PRODOTTO ALLA LISTA(@ AddElement)----**

INSERT INTO prodotto (ListaDesideriID, Nome) VALUES ('1','Mango');

**---- ELIMINAZIONE PRODOTTO DALLA LISTA (@RemoveElement) ----**

DELETE FROM prodotto WHERE ID='5';

---- @RenameList

Non la facciamo

**---- @RemoveList ----**

DELETE FROM listadesideri AS lista WHERE lista.ID = '3';

**### GRUPPO ###**

**---- SELEZIONA I PARTECIPANTI E LE LISTE CONDIVISE DI UN GRUPPO (@ GetGroup) ----**

SELECT g.ID AS "ID gruppo", par.ID AS "ID part. al gruppo", par.UtenteProprietario AS "E' proprietario del gruppo", u.ID AS "ID utente", u.Nome, u.Cognome, gl.ListaDesideriID, lista.Nome AS "Nome Lista"

FROM utente AS u

INNER JOIN partecipazione AS par ON u.ID = par.UtenteID

INNER JOIN gruppo AS g ON par.GruppoID = g.ID AND g.ID = '2'

INNER JOIN gruppo\_listadesideri AS gl ON g.ID = gl.GruppoID AND g.ID = '2'

INNER JOIN listadesideri AS lista ON gl.ListaDesideriID = lista.ID

WHERE par.UtenteID = lista.UtenteID;

---- @RenameGroup

Non la facciamo

**---- @RemoveGroup ----**

DELETE FROM gruppo AS g WHERE g.ID = '3';

**----** (@AddPartecipant)

Non la facciamo

**---- ELIMINAZIONE DI PARTECIPAZIONE [ATTENZIONE TRIGGER] (@DecouplePartecipant) ----**

DELETE FROM partecipazione AS par WHERE par.GruppoID = '2' AND par.UtenteID = '3';

CREATE TRIGGER after\_delete\_partecipazione

AFTER DELETE

ON partecipazione FOR EACH ROW

DELETE FROM gruppo\_listadesideri AS gl WHERE gl.GruppoID = OLD.GruppoID

AND EXISTS(SELECT \* FROM listadesideri AS lista WHERE gl.ListaDesideriID = lista.ID AND lista.UtenteID = OLD.UtenteID);

**---- @DecoupleList [ATTENZIONE TRIGGER]**

DELETE FROM listadesideri AS lista WHERE lista.ID = '7';

DELIMITER $$

CREATE TRIGGER after\_delete\_lista

AFTER DELETE

ON listadesideri FOR EACH ROW

BEGIN

DELETE FROM prodotto AS p WHERE p.ListaDesideriID = OLD.ID;

DELETE FROM gruppo\_listadesideri AS gl WHERE gl.ListaDesideriID = OLD.ID;

END $$

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`progettossd`.`gruppo\_listadesideri`, CONSTRAINT `FKGruppo\_Lis683055` FOREIGN KEY (`ListaDesideriID`) REFERENCES `listadesideri` (`ID`))

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`progettossd`.`prodotto`, CONSTRAINT `FKProdotto471876` FOREIGN KEY (`ListaDesideriID`) REFERENCES `listadesideri` (`ID`))

**#### DB-ATTRIBUTE-FINDER ####**

**---- @function\_id\_prop\_lista**

SELECT UtenteID FROM listadesideri AS lista WHERE lista.ID = '3';

**---- @function\_id\_liste\_cond**

SELECT ListaDesideriID FROM gruppo\_listadesideri AS gl WHERE gl.GruppoID = '1';

---- @function\_id\_prop\_gruppo

SELECT UtenteID FROM partecipazione AS p WHERE p.GruppoID = '1' and p.UtenteProprietario = '1';

----- function\_id\_part\_gruppo

SELECT UtenteID FROM partecipazione AS p WHERE p.GruppoID = '2';

**PER MODIFICARE ON DELETE CASCADE ETC.**

ALTER TABLE `progettossd`.`gruppo\_listadesideri`

DROP FOREIGN KEY `FKGruppo\_Lis633240`;

ALTER TABLE `progettossd`.`gruppo\_listadesideri`

ADD CONSTRAINT `FKGruppo\_Lis633240`

FOREIGN KEY (`GruppoID`)

REFERENCES `progettossd`.`gruppo` (`ID`)

ON DELETE CASCADE

ON UPDATE RESTRICT;

[poi stessa cosa anche per altra foreign key di gruppo\_listadesideri]

ALTER TABLE `progettossd`.`prodotto`

DROP FOREIGN KEY `FKProdotto471876`;

ALTER TABLE `progettossd`.`prodotto`

ADD CONSTRAINT `FKProdotto471876`

FOREIGN KEY (`ListaDesideriID`)

REFERENCES `progettossd`.`listadesideri` (`ID`)

ON DELETE CASCADE;

TRIGGER SBAGLIATO?

CREATE TRIGGER after\_insert\_gruppo

AFTER INSERT

ON gruppo FOR EACH ROW

INSERT INTO partecipazione(UtenteProprietario, UtenteID, GruppoID) VALUES ('1',);

QUERY SBAGLIATE

SELECT \* FROM utente AS u WHERE EXISTS (SELECT GruppoID FROM partecipazione AS par WHERE u.ID = par.UtenteID AND par.GruppoID = '2');

SELECT \* FROM listadesideri AS lista WHERE EXISTS (SELECT GruppoID FROM gruppo\_listadesideri AS gl WHERE lista.ID = gl.ListaDesideriID AND gl.GruppoID = '2');

* SELECT Nome FROM gruppo AS g WHERE g.ID = '1';

SELECT g.ID AS "id gruppo", par.ID AS "ID part. al gruppo", u.ID AS "ID utente", u.Nome, u.Cognome, par.UtenteProprietario AS "E' proprietario", gl.ListaDesideriID, lista.Nome AS "Nome Lista"

FROM utente AS u

INNER JOIN partecipazione AS par ON u.ID = par.UtenteID

INNER JOIN gruppo AS g ON par.GruppoID = g.ID AND g.ID = '2'

INNER JOIN gruppo\_listadesideri AS gl ON g.ID = gl.GruppoID AND g.ID = '2'

INNER JOIN listadesideri AS lista ON gl.ListaDesideriID = lista.ID;

SELECT g.ID AS "id gruppo", par.ID AS "ID part. al gruppo", par.UtenteProprietario AS "E' proprietario del gruppo", gl.ListaDesideriID AS "LISTA ID"

FROM partecipazione AS par

INNER JOIN gruppo AS g ON par.GruppoID = g.ID AND g.ID = '2'

INNER JOIN gruppo\_listadesideri AS gl ON g.ID = gl.GruppoID

INNER JOIN listadesideri AS lista ON gl.ListaDesideriID = lista.ID

INNER JOIN utente AS u ON lista.UtenteID = u.ID

WHERE par.UtenteID = lista.UtenteID;