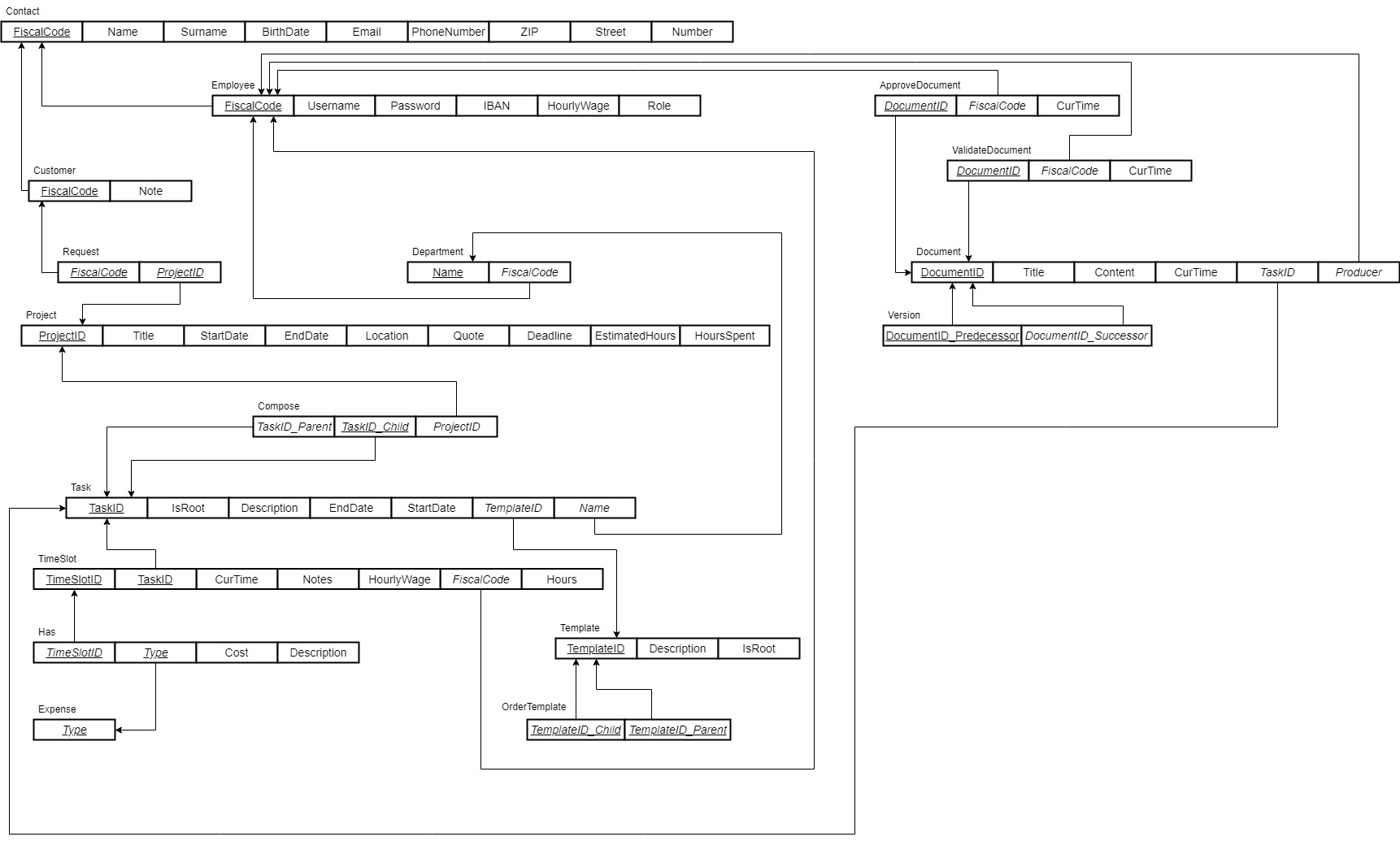
Variations to the Relational Schema

Figure 1 shows the relational schema. The type of the attribute ‘EstimatedHours’ in the relation Project has been changed from ‘Float’ to ‘Integer’ because the information does not need to be that precise.

*Figure 1: Relational Schema*

Physical Schema

In the following the SQL instructions to build the database in Figure 1 are reported. Note that the tables should be created in the correct order, as reported thereafter.

-- Database Creation

CREATE DATABASE ennedue OWNER POSTGRES ENCODING = 'UTF8';

-- Connect to examode db to create data for its 'public' schema

\c ennedue

-- Install the extention for the uuid: uuid-ossp.

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Create new domains

CREATE DOMAIN pwd AS character varying(254)

CONSTRAINT properpassword CHECK (((VALUE)::text ~\* '[A-Za-z0-9.\_%-]{5,}'::text));

-- Create new data types

CREATE TYPE roleType AS ENUM (

'Employee',

'Chief',

'Area Manager'

);

-- Table Creation

-- FiscalCode has VARCHAR(16) because the fiscal code length is always 16

-- Contact

CREATE TABLE Contact(

FiscalCode VARCHAR(16),

Name VARCHAR NOT NULL,

Surname VARCHAR NOT NULL,

BirthDate DATE,

Email VARCHAR,

PhoneNumber VARCHAR NOT NULL,

ZIP VARCHAR(5) NOT NULL,

Street VARCHAR,

Number SMALLINT,

CONSTRAINT Address CHECK ((Street is NULL AND Number is NULL) OR (Street is NOT NULL AND Number is NOT NULL)),

PRIMARY KEY (FiscalCode)

);

COMMENT ON TABLE Contact IS 'Represents a contact.';

COMMENT ON COLUMN Contact.FiscalCode IS 'The unique fiscalCode of the contact.';

COMMENT ON COLUMN Contact.Name IS 'The name of the contact.';

COMMENT ON COLUMN Contact.Surname IS 'The surname of the contact.';

COMMENT ON COLUMN Contact.BirthDate IS 'The birth date of the contact.';

COMMENT ON COLUMN Contact.Email IS 'The email of the contact.';

COMMENT ON COLUMN Contact.PhoneNumber IS 'The phone number of the contact.';

COMMENT ON COLUMN Contact.ZIP IS 'The Zone Improvement Plan of the contact.';

COMMENT ON COLUMN Contact.Street IS 'The street of the contact.';

COMMENT ON COLUMN Contact.Number IS 'The address number of the contact.';

-- Employee

CREATE TABLE Employee(

FiscalCode VARCHAR(16),

Username VARCHAR NOT NULL UNIQUE,

Password VARCHAR NOT NULL,

IBAN VARCHAR NOT NULL,

HourlyWage float(2) NOT NULL,

Role roleType NOT NULL,

PRIMARY KEY (FiscalCode),

FOREIGN KEY (FiscalCode) REFERENCES Contact(FiscalCode)

);

COMMENT ON TABLE Employee IS 'Represents an employee.';

COMMENT ON COLUMN Employee.FiscalCode IS 'The unique fiscalCode of the employee.';

COMMENT ON COLUMN Employee.Username IS 'The username for the login of the employee.';

COMMENT ON COLUMN Employee.Password IS 'The password for the login of the employee.';

COMMENT ON COLUMN Employee.IBAN IS 'The IBAN of the employee.';

COMMENT ON COLUMN Employee.HourlyWage IS 'The hourly wage of the employee.';

COMMENT ON COLUMN Employee.Role IS 'The role in the studio of the employee.';

-- Customer

CREATE TABLE Customer(

FiscalCode VARCHAR(16),

Note TEXT,

PRIMARY KEY (FiscalCode),

FOREIGN KEY (FiscalCode) REFERENCES Contact(FiscalCode)

);

COMMENT ON TABLE Customer IS 'Represents a customer.';

COMMENT ON COLUMN Customer.FiscalCode IS 'The unique FiscalCode of the customer.';

COMMENT ON COLUMN Customer.Note IS 'Some annotations about the customer.';

-- Project

CREATE TABLE Project(

ProjectID UUID,

Title VARCHAR NOT NULL,

StartDate DATE NOT NULL,

EndDate DATE,

Location VARCHAR NOT NULL,

Quote BYTEA,

Deadline DATE NOT NULL,

EstimatedHours INTEGER NOT NULL,

HoursSpent FLOAT(2) DEFAULT 0.00,

PRIMARY KEY (ProjectID)

);

COMMENT ON TABLE Project IS 'Represents a project.';

COMMENT ON COLUMN Project.ProjectID IS 'The unique ID of the project.';

COMMENT ON COLUMN Project.Title IS 'The title of the project.';

COMMENT ON COLUMN Project.StartDate IS 'The start date of the project';

COMMENT ON COLUMN Project.EndDate IS 'The end date of the project';

COMMENT ON COLUMN Project.Location IS 'The location of the project';

COMMENT ON COLUMN Project.Quote IS 'The quote of the project';

COMMENT ON COLUMN Project.Deadline IS 'The deadline of the project';

COMMENT ON COLUMN Project.EstimatedHours IS 'The estimated hours for the project';

COMMENT ON COLUMN Project.HoursSpent IS 'The hours spent for the project';

-- Request

CREATE TABLE Request(

FiscalCode VARCHAR(16),

ProjectID UUID,

PRIMARY KEY (FiscalCode, ProjectID),

FOREIGN KEY (FiscalCode) REFERENCES Customer(FiscalCode),

FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)

);

COMMENT ON TABLE Request IS 'Represents the request for a project.';

COMMENT ON COLUMN Request.FiscalCode IS 'The unique fiscal code of the customer.';

COMMENT ON COLUMN Request.ProjectID IS 'The unique ID of the requested project.';

-- Department

CREATE TABLE Department(

Name VARCHAR(50),

FiscalCode VARCHAR(16) NOT NULL,

PRIMARY KEY (Name),

FOREIGN KEY (FiscalCode) REFERENCES Employee(FiscalCode)

);

COMMENT ON TABLE Department IS 'Represents a specific area of the studio.';

COMMENT ON COLUMN Department.Name IS 'The name of the department.';

COMMENT ON COLUMN Department.FiscalCode IS 'The fiscal code of the employee who leads the department.';

-- Template

CREATE TABLE Template(

TemplateID VARCHAR,

Description TEXT,

IsRoot BOOLEAN NOT NULL,

PRIMARY KEY (TemplateID)

);

COMMENT ON TABLE Template IS 'Represents a set of predefined templates.';

COMMENT ON COLUMN Template.TemplateID IS 'The name of the template.';

COMMENT ON COLUMN Template.Description IS 'The description of the template.';

COMMENT ON COLUMN Template.IsRoot IS 'Determines if the template is the root of the tree.';

-- Order

CREATE TABLE OrderTemplate(

Child VARCHAR,

Parent VARCHAR,

PRIMARY KEY (Child, Parent),

FOREIGN KEY (Child) REFERENCES Template(TemplateID),

FOREIGN KEY (Parent) REFERENCES Template(TemplateID)

);

COMMENT ON TABLE OrderTemplate IS 'Handles the order of the template.';

COMMENT ON COLUMN OrderTemplate.Child IS 'The identifier of the child node.';

COMMENT ON COLUMN OrderTemplate.Parent IS 'The identifier of the parent node.';

-- Task

CREATE TABLE Task(

TaskID UUID,

IsRoot BOOLEAN NOT NULL,

Description TEXT,

StartDate DATE NOT NULL,

EndDate DATE,

TemplateID VARCHAR NOT NULL,

Name VARCHAR(50) NOT NULL,

PRIMARY KEY (TaskID),

FOREIGN KEY (TemplateID) REFERENCES Template(TemplateID),

FOREIGN KEY (Name) REFERENCES Department(Name)

);

COMMENT ON TABLE Task IS 'Represents an activity of the studio.';

COMMENT ON COLUMN Task.TaskID IS 'The unique ID of the task.';

COMMENT ON COLUMN Task.IsRoot IS 'Determines if the task is the root of the tree.';

COMMENT ON COLUMN Task.Description IS 'The description of the task.';

COMMENT ON COLUMN Task.StartDate IS 'The start date of the task.';

COMMENT ON COLUMN Task.EndDate IS 'The end date of the task.';

COMMENT ON COLUMN Task.TemplateID IS 'The name of the template.';

COMMENT ON COLUMN Task.Name IS 'The name of the department associated to the task.';

-- Compose

CREATE TABLE Compose(

Parent UUID,

Child UUID,

ProjectID UUID NOT NULL,

PRIMARY KEY (Child),

FOREIGN KEY (Parent) REFERENCES Task(TaskID),

FOREIGN KEY (Child) REFERENCES Task(TaskID),

FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)

);

COMMENT ON TABLE Compose IS 'Represents which tasks compose a project.';

COMMENT ON COLUMN Compose.Parent IS 'The identifier of the parent task.';

COMMENT ON COLUMN Compose.Child IS 'The identifier of the child task.';

COMMENT ON COLUMN Compose.ProjectID IS 'The unique identifier of the associated project.';

-- TimeSlot

CREATE TABLE TimeSlot(

TimeSlotID UUID,

TaskID UUID,

CurTime TIMESTAMPTZ NOT NULL,

FiscalCode VARCHAR(16) NOT NULL,

Notes TEXT,

Hours float(2) NOT NULL,

HourlyWage float(2) NOT NULL,

PRIMARY KEY(TimeSlotID),

FOREIGN KEY(TaskID) REFERENCES Task(TaskID),

FOREIGN KEY(FiscalCode) REFERENCES Employee(FiscalCode)

);

COMMENT ON TABLE TimeSlot IS 'Represents the contribute of each employee in different fractions of time.';

COMMENT ON COLUMN TimeSlot.TimeSlotID IS 'The unique identifier of a specific time slot.';

COMMENT ON COLUMN TimeSlot.TaskID IS 'The unique identifier of the task.';

COMMENT ON COLUMN TimeSlot.CurTime IS 'The date and time when a time slot has been reported.';

COMMENT ON COLUMN TimeSlot.FiscalCode IS 'The unique fiscal code of the employee.';

COMMENT ON COLUMN TimeSlot.Notes IS 'Notes about the time slot.';

COMMENT ON COLUMN TimeSlot.Hours IS 'The number of hours spent in the time slot.';

COMMENT ON COLUMN TimeSlot.HourlyWage IS 'The employee’s wage per hour.';

-- Expense

CREATE TABLE Expense(

Type VARCHAR,

PRIMARY KEY (Type)

);

COMMENT ON TABLE Expense IS 'Represents the expense incurred by an employee that has to be reimbursed.';

COMMENT ON COLUMN Expense.Type IS 'The type of expense.';

-- Has

CREATE TABLE Has(

TimeSlotID UUID,

Type VARCHAR,

Cost float(2) NOT NULL,

Description TEXT,

PRIMARY KEY (TimeSlotID, Type),

FOREIGN KEY (TimeSlotID) REFERENCES Timeslot(TimeSlotID),

FOREIGN KEY (Type) REFERENCES Expense (Type)

);

COMMENT ON TABLE Has IS 'Represents the featurse of an expense.';

COMMENT ON COLUMN Has.TimeSlotID IS 'The unique identifier of a specific time slot.';

COMMENT ON COLUMN Has.Type IS 'The type of expense.';

COMMENT ON COLUMN Has.Cost IS 'The amount of the expense.';

COMMENT ON COLUMN Has.Description IS 'The description of the time slot.';

-- Document

CREATE TABLE Document(

DocumentID UUID,

Title VARCHAR NOT NULL,

Content BYTEA NOT NULL,

CurTime TIMESTAMPTZ NOT NULL,

TaskID UUID NOT NULL,

Producer VARCHAR(16),

PRIMARY KEY (DocumentID),

FOREIGN KEY (TaskID) REFERENCES Task(TaskID),

FOREIGN KEY (Producer) REFERENCES Employee(FiscalCode)

);

COMMENT ON TABLE Document IS 'Represents the document generated by the studio.';

COMMENT ON COLUMN Document.DocumentID IS 'The unique identifier of the document.';

COMMENT ON COLUMN Document.Title IS 'The title of the document.';

COMMENT ON COLUMN Document.Content IS 'The content of the document.';

COMMENT ON COLUMN Document.CurTime IS 'The date when a document has been received.';

COMMENT ON COLUMN Document.TaskID IS 'The unique identifier of the task.';

COMMENT ON COLUMN Document.Producer IS 'The producer of the document.';

-- Version

CREATE TABLE Version(

Predecessor UUID,

Successor UUID NOT NULL,

PRIMARY KEY (Predecessor),

FOREIGN KEY (Predecessor) REFERENCES Document(DocumentID),

FOREIGN KEY (Successor) REFERENCES Document(DocumentID)

);

COMMENT ON TABLE Version IS 'Represents the version of the document.';

COMMENT ON COLUMN Version.Predecessor IS 'The unique identifier of the previous version of the document.';

COMMENT ON COLUMN Version.Successor IS 'The unique identifier of the updated version of the document.';

-- Validate

create TABLE ValidateDocument(

DocumentID UUID,

FiscalCode VARCHAR(16) NOT NULL,

CurTime TIMESTAMPTZ NOT NULL,

PRIMARY KEY (DocumentID),

FOREIGN KEY (FiscalCode) REFERENCES Employee(FiscalCode),

FOREIGN KEY (DocumentID) REFERENCES Document(DocumentID)

);

COMMENT ON TABLE ValidateDocument IS 'Represents the validation of a document.';

COMMENT ON COLUMN ValidateDocument.DocumentID IS 'The unique identifier of the document.';

COMMENT ON COLUMN ValidateDocument.FiscalCode IS 'The unique fiscal code of the employee.';

COMMENT ON COLUMN ValidateDocument.CurTime IS 'The date when a document has been validated.';

-- Approve

create TABLE ApproveDocument(

DocumentID UUID,

FiscalCode VARCHAR(16) NOT NULL,

CurTime TIMESTAMPTZ NOT NULL,

PRIMARY KEY (DocumentID),

FOREIGN KEY (FiscalCode) REFERENCES Employee(FiscalCode),

FOREIGN KEY (DocumentID) REFERENCES Document(DocumentID)

);

COMMENT ON TABLE ApproveDocument IS 'Represents the approvation of a document.';

COMMENT ON COLUMN ApproveDocument.DocumentID IS 'The unique identifier of the document.';

COMMENT ON COLUMN ApproveDocument.FiscalCode IS 'The unique fiscal code of the employee.';

COMMENT ON COLUMN ApproveDocument.CurTime IS 'The date when a document has been approved.';

Trigger Function

The first trigger checks if there is correspondence between the hourly wage inside a new timeslot and the actual hourly wage associated to the employee. The second trigger guarantees the data integrity between the hours reported by each employee and the total hours spent on a specific project, in fact when inserting or updating a timeslot the trigger updates the hours counter of the project associated to the employee’s report.

-- Connect to the database ennedue

\c ennedue

-- When an Employee reports a certain Task, the HourlyWage declared in his dedicated Timeslot must be consistent with the one assigned to him in the Employee relation.

CREATE FUNCTION checkHourlyWage() RETURNS TRIGGER AS $$

BEGIN

-- Join the Employee table with the TimeSlot table

-- Check if the new HourlyWage inserted in TimeSlot is different from the one related to the associated Employee

IF NOT NEW.HourlyWage IN (SELECT E.HourlyWage

FROM Employee AS E INNER JOIN TimeSlot AS TS

ON E.FiscalCode = TS.FiscalCode

WHERE TS.TaskID = NEW.TaskID) THEN

-- If not, the new TimeSlot data cannot be inserted.

RAISE EXCEPTION 'Inconsistent Hourly Wage %.', NEW.HourlyWage USING HINT = 'Please check correctness';

END IF;

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER CheckWage

AFTER INSERT ON TimeSlot

FOR EACH ROW

EXECUTE PROCEDURE checkHourlyWage();

-- When an Employee inserts new data into the TimeSlot table, the total amout of HoursSpent for the Project must be updated by adding the new ones.

CREATE FUNCTION updateHoursSpent() RETURNS TRIGGER AS $$

BEGIN

-- Update the Project table

UPDATE Project

-- Sum the total amount of hours with the inserted ones

SET HoursSpent = Project.HoursSpent + NEW.Hours

-- Join TimeSlot, Task, Compose and Project tables and select the correct project from the Project table.

WHERE ProjectID IN (SELECT P.ProjectID

FROM Task AS T INNER JOIN TimeSlot AS TS

ON T.TaskID = TS.TaskID

INNER JOIN Compose AS C

ON TS.TaskID = C.Child

INNER JOIN Project AS P

ON C.ProjectID = P.ProjectID

WHERE C.Child = NEW.taskid);

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER UpdateHours

AFTER INSERT OR UPDATE ON TimeSlot

FOR EACH ROW

EXECUTE PROCEDURE updateHoursSpent();

Populate the Database: Example

In the following, there are some examples of SQL instructions to insert data within the main relations of the database. Below it is also reported the Java code for inserting PDFs inside the database and also for retrieving them from the DBMS.

-- Connect to the database ennedue

\c ennedue

--Insert Operations

--Contact

INSERT INTO Contact(FiscalCode, Name, Surname, BirthDate, Email, PhoneNumber, ZIP, Street, Number) VALUES

('MNGSLV89H60Z100C','Silvia','Mengotti','1989-08-20','silviamengotti@gmail.com','3498975114','35017','Via Piave',27),

('LRNBTT96C01D149A','Lorenzo','Bottiglia','1996-03-01','lorenzo01@gmail.com','3458975211','36061','Via San Giuseppe ',1),

('RSSNRD85D05A044I','Andrea','Rossi','1985-04-05',NULL,'3281515616','30012','Via II Febbraio',12),

('RSSGVN76A15E189E','Giovanni','Rossi','1976-01-15','giogio@libero.it','3494949122','36061','Via Santuario',16),

('BRNGNN64G46H005D','Gianna','Bruni','1964-07-06',NULL,'3271231112','34815','Via Roma',37),

('VLNLIA88A01A007M','Elia','Valencia','1988-01-01','elia88@hotmail.com','3251516878','36116','Via Verci',22),

('BRDSRA77C43M401T','Sara','Bardi','1977-03-03','sara0303@hotmail.com','3411579852','36061','Via Marinali',14),

('RSSMRA70A01L726S','Mario','Rossi','1970-01-01','sigrossi@hotmail.com','3271818523','35082','Via Costa',11),

('PVNTHM56M14A001M','Thomas','Pavan','1956-10-14',NULL,'3448972123','34014','Via Adriatico',46);

--Employee

INSERT INTO Employee(FiscalCode, Username, Password, IBAN, HourlyWage, Role) VALUES

('LRNBTT96C01D149A','bottiglialore','1996lorenzo','IT96Z9645879900115684001800',6.50,'Employee'),

('RSSGVN76A15E189E','rossigio76','giovanni1976','IT46A0258760055890317000011',7.80,'Employee'),

('BRNGNN64G46H005D','brunigianna46','missbruni46','IT01M0700001145620000002315',12.50,'Chief'),

('MNGSLV89H60Z100C','mengottisilvia100','silvia1234.','IT47A0789560000002516182022',9.30,'Area Manager');

--Customer

INSERT INTO Customer(FiscalCode,Note) VALUES

('RSSMRA70A01L726S',NULL),

('PVNTHM56M14A001M','Good Customer'),

('RSSNRD85D05A044I','Problematic payment issues'),

('BRDSRA77C43M401T','Very polite and punctual');

--Project

INSERT INTO Project(ProjectID, Title, StartDate, EndDate, Location, Quote, Deadline, EstimatedHours, HoursSpent) VALUES -- BETA TEST VALUES FOR EstimatedHours (NEED UPDATE)

('de7c222e-98f0-4eae-b690-7fb37a246bdd','The New House','2018-07-25','2018-08-25','Padova','300','2018-08-30',112,94.00),

('757ca527-b338-42f5-bbe2-1075d63b492c','Bathroom','2018-08-10','2018-10-17','Roma','160','2018-10-20',80,45.00),

('866adc89-5a04-463b-82a3-3bf0ad77491b','Company Building','2019-05-10',NULL,'Venezia','230','2020-01-12',1600,1000.00);

--Request

INSERT INTO Request(FiscalCode, ProjectID) VALUES

('PVNTHM56M14A001M','757ca527-b338-42f5-bbe2-1075d63b492c'),

('BRDSRA77C43M401T','866adc89-5a04-463b-82a3-3bf0ad77491b'),

('RSSMRA70A01L726S','de7c222e-98f0-4eae-b690-7fb37a246bdd');

--Department

INSERT INTO Department(Name, FiscalCode) VALUES

('PlanificationDepartment', 'RSSGVN76A15E189E'),

('VeficationDepartment', 'LRNBTT96C01D149A'),

('ProjectingDepartment', 'MNGSLV89H60Z100C');

--Template

INSERT INTO Template(TemplateID, Description, IsRoot) VALUES

('concept', 'Concept design for the project.', '1'),

('sketches','Sketches with drawings.', '0'),

('architectural model', 'Model of the hourse.', '0'),

('realization', 'Creation of the final drawing.', '0'),

('architectural executive', 'Realization of the project.', '1'),

('task assignment', 'Organize and assign the task to employees.', '0'),

('details analysis', 'Get all the measures for the realization of the project.', '0'),

('draft', 'Create a simple concept of the project.', '0'),

('drawing', '2D Drawing.', '0'),

('computations', 'Calculus.', '0'),

('modelling', 'Create a model for the project.', '0'),

('take-over', 'Get all the necessary informations for the project.', '0');

--OrderTemplate

INSERT INTO OrderTemplate(Child, Parent) VALUES

('sketches','concept'),

('architectural model','concept'),

('realization','concept'),

('task assignment', 'architectural executive'),

('details analysis','architectural executive'),

('draft','architectural executive'),

('drawing','sketches'),

('computations','sketches'),

('modelling','sketches'),

('take-over','sketches'),

('drawing','architectural model'),

('computations','architectural model'),

('modelling','architectural model'),

('take-over','architectural model'),

('drawing','realization'),

('computations','realization'),

('modelling','realization'),

('take-over','realization'),

('drawing','task assignment'),

('computations','task assignment'),

('modelling','task assignment'),

('take-over','task assignment'),

('drawing','details analysis'),

('computations','details analysis'),

('modelling','details analysis'),

('take-over','details analysis'),

('drawing','draft'),

('computations','draft'),

('modelling','draft'),

('take-over','draft');

--Task

--For the first operation insert, I have considered a template and I have specified that the specified task need to be done for that template

INSERT INTO Task(TaskID, IsRoot, Description, EndDate, StartDate, TemplateID, Name) VALUES

--Project 1

('c769d3a6-41d1-4883-9edf-e74a977446ad','1','The living room must have a large open space.','2018-08-25','2018-07-25','concept','PlanificationDepartment'),

('f77d1e8b-3b5f-491c-987c-8d5d77baba3a','0',NULL,'2018-08-03','2018-07-25','sketches','PlanificationDepartment'),

('e527d149-b101-4bbb-b86f-29ca2ccf6b99','0',NULL,'2018-08-03','2018-07-25','drawing','PlanificationDepartment'),

('353bf9d0-183e-469d-978a-9484f6c25b15','0',NULL,'2018-08-03','2018-07-25','modelling','PlanificationDepartment'),

('fc63b2ab-4aa3-45e7-a5a0-85f1436f81b2','0',NULL,'2018-08-25','2018-08-03','architectural model','PlanificationDepartment'),

('7632306e-63eb-4a48-a630-8891a06580a8','0',NULL,'2018-08-25','2018-08-03','modelling','PlanificationDepartment'),

--Project 2

('bb6a4192-90d6-4f97-ab66-6518d29a3537','1','This task needs to be completed 1 week before deadline.','2018-10-17','2018-08-10','architectural executive','ProjectingDepartment'),

('f23af6d9-6f21-42fd-b35b-e8f9bbc8a753','0',NULL,'2018-10-03','2018-08-10','task assignment','ProjectingDepartment'),

('fe081609-5c64-465c-b96e-e3c7668d72bc','0',NULL,'2018-10-03','2018-08-10','computations','ProjectingDepartment'),

('056d76ff-01c6-4248-9749-8a36a26a1144','0',NULL,'2018-10-03','2018-08-10','take-over','ProjectingDepartment'),

('197dd0f2-56f3-449b-9241-92491a773e5e','0',NULL,'2018-10-10','2018-10-03','draft','ProjectingDepartment'),

('735dcc1b-2843-41de-a74c-c040d6f8155b','0',NULL,'2018-10-10','2018-10-03','drawing','ProjectingDepartment'),

('d17c7007-fa6a-4cbb-bc72-9d51fd6c1796','0',NULL,'2018-10-10','2018-10-03','computations','ProjectingDepartment'),

--Project 3

('e6f40259-69f5-402c-9616-dc58fbd1fb4b','1','Task to be completed carefully.',NULL,'2019-05-10','concept','VeficationDepartment'),

('84a0ee99-4702-41de-8aff-185d257c9dc1','0',NULL,NULL,'2019-05-10','realization','VeficationDepartment'),

('165b25b4-8b44-46eb-b8ca-4becc7e6ca7c','0',NULL,NULL,'2019-05-10','drawing','VeficationDepartment'),

('35e0ef16-edee-4a30-9f46-80b0b959610c','0',NULL,NULL,'2019-05-10','computations','VeficationDepartment'),

('644932c2-c6c5-4872-b6a8-114634c6a472','0',NULL,NULL,'2019-05-10','modelling','VeficationDepartment'),

('850216a6-3c23-4aca-ad7d-ee0b1916bc7b','0',NULL,'2019-05-25','2019-05-10','take-over','VeficationDepartment');

--Compose

INSERT INTO Compose(Parent, Child, ProjectID) VALUES

--Project 1

(NULL,'c769d3a6-41d1-4883-9edf-e74a977446ad','de7c222e-98f0-4eae-b690-7fb37a246bdd'),

('c769d3a6-41d1-4883-9edf-e74a977446ad','f77d1e8b-3b5f-491c-987c-8d5d77baba3a','de7c222e-98f0-4eae-b690-7fb37a246bdd'),

('f77d1e8b-3b5f-491c-987c-8d5d77baba3a','e527d149-b101-4bbb-b86f-29ca2ccf6b99','de7c222e-98f0-4eae-b690-7fb37a246bdd'),

('f77d1e8b-3b5f-491c-987c-8d5d77baba3a','353bf9d0-183e-469d-978a-9484f6c25b15','de7c222e-98f0-4eae-b690-7fb37a246bdd'),

('c769d3a6-41d1-4883-9edf-e74a977446ad','fc63b2ab-4aa3-45e7-a5a0-85f1436f81b2','de7c222e-98f0-4eae-b690-7fb37a246bdd'),

('fc63b2ab-4aa3-45e7-a5a0-85f1436f81b2','7632306e-63eb-4a48-a630-8891a06580a8','de7c222e-98f0-4eae-b690-7fb37a246bdd'),

--Project 2

(NULL,'bb6a4192-90d6-4f97-ab66-6518d29a3537','757ca527-b338-42f5-bbe2-1075d63b492c'),

('bb6a4192-90d6-4f97-ab66-6518d29a3537','f23af6d9-6f21-42fd-b35b-e8f9bbc8a753','757ca527-b338-42f5-bbe2-1075d63b492c'),

('f23af6d9-6f21-42fd-b35b-e8f9bbc8a753','fe081609-5c64-465c-b96e-e3c7668d72bc','757ca527-b338-42f5-bbe2-1075d63b492c'),

('f23af6d9-6f21-42fd-b35b-e8f9bbc8a753','056d76ff-01c6-4248-9749-8a36a26a1144','757ca527-b338-42f5-bbe2-1075d63b492c'),

('bb6a4192-90d6-4f97-ab66-6518d29a3537','197dd0f2-56f3-449b-9241-92491a773e5e','757ca527-b338-42f5-bbe2-1075d63b492c'),

('197dd0f2-56f3-449b-9241-92491a773e5e','735dcc1b-2843-41de-a74c-c040d6f8155b','757ca527-b338-42f5-bbe2-1075d63b492c'),

('197dd0f2-56f3-449b-9241-92491a773e5e','d17c7007-fa6a-4cbb-bc72-9d51fd6c1796','757ca527-b338-42f5-bbe2-1075d63b492c'),

--Project 3

(NULL,'e6f40259-69f5-402c-9616-dc58fbd1fb4b','866adc89-5a04-463b-82a3-3bf0ad77491b'),

('e6f40259-69f5-402c-9616-dc58fbd1fb4b','84a0ee99-4702-41de-8aff-185d257c9dc1','866adc89-5a04-463b-82a3-3bf0ad77491b'),

('84a0ee99-4702-41de-8aff-185d257c9dc1','165b25b4-8b44-46eb-b8ca-4becc7e6ca7c','866adc89-5a04-463b-82a3-3bf0ad77491b'),

('84a0ee99-4702-41de-8aff-185d257c9dc1','35e0ef16-edee-4a30-9f46-80b0b959610c','866adc89-5a04-463b-82a3-3bf0ad77491b'),

('84a0ee99-4702-41de-8aff-185d257c9dc1','644932c2-c6c5-4872-b6a8-114634c6a472','866adc89-5a04-463b-82a3-3bf0ad77491b'),

('84a0ee99-4702-41de-8aff-185d257c9dc1','850216a6-3c23-4aca-ad7d-ee0b1916bc7b','866adc89-5a04-463b-82a3-3bf0ad77491b');

--TimeSlot

--The hours specified in this insertions are added to the total amount of hours dedicated to the project by means of the UpdateHours trigger

INSERT INTO TimeSlot(TimeSlotID, TaskID, CurTime, Notes, HourlyWage, FiscalCode, Hours) VALUES

--Project 1

('69aed574-6572-42f0-863e-c7ba2260d752','353bf9d0-183e-469d-978a-9484f6c25b15','2018-07-25 16:32:25+01',NULL,'6.50','LRNBTT96C01D149A','5.5'),

('e5b6ca51-14e6-4959-b0c4-e9d882a2fcde','353bf9d0-183e-469d-978a-9484f6c25b15','2018-07-28 13:32:25+01',NULL,'6.50','LRNBTT96C01D149A','6.0'),

('6f4fc82d-0e6c-4395-85ab-e6f919620f93','353bf9d0-183e-469d-978a-9484f6c25b15','2018-07-30 18:32:25+01',NULL,'6.50','LRNBTT96C01D149A','18.5'),

('b0426e09-1097-44db-9a55-113c9330ab1e','e527d149-b101-4bbb-b86f-29ca2ccf6b99','2018-08-01 11:32:25+01',NULL,'9.30','MNGSLV89H60Z100C','9.0'),

('ee6b4167-1c71-4b16-9f9d-9ddbeda67c1c','7632306e-63eb-4a48-a630-8891a06580a8','2018-08-24 10:32:25+01',NULL,'9.30','MNGSLV89H60Z100C','4.0'),

--Project 2

('12476d9b-058c-4a71-b33b-bdca507e53c6','fe081609-5c64-465c-b96e-e3c7668d72bc','2018-08-11 16:32:25+01',NULL,'7.80','RSSGVN76A15E189E','2.5'),

('8563d0ef-a842-42c6-b613-b9bf824977bc','056d76ff-01c6-4248-9749-8a36a26a1144','2018-09-03 12:32:25+01',NULL,'7.80','RSSGVN76A15E189E','2.5'),

('976af0cc-ceee-4477-b81e-cf0199bb07bf','735dcc1b-2843-41de-a74c-c040d6f8155b','2018-10-03 14:32:25+01',NULL,'7.80','RSSGVN76A15E189E','6.0'),

('a1001926-a9cf-4b4e-98fa-96ee0d546187','d17c7007-fa6a-4cbb-bc72-9d51fd6c1796','2018-10-09 22:32:25+01',NULL,'7.80','RSSGVN76A15E189E','7.0'),

--Project 3

('31f1de04-e117-4463-8eef-ba0598502e66', '165b25b4-8b44-46eb-b8ca-4becc7e6ca7c','2018-05-10 10:32:25+01',NULL,'12.50','BRNGNN64G46H005D','3.5'),

('1a14cdb1-6d78-4266-b9ea-7e76d8822701', '35e0ef16-edee-4a30-9f46-80b0b959610c','2018-05-14 12:32:25+01',NULL,'12.50','BRNGNN64G46H005D','10.5'),

('2cae32d3-7bf1-435f-b851-9220fb8c2ca1', '644932c2-c6c5-4872-b6a8-114634c6a472','2018-05-17 18:32:25+01',NULL,'12.50','BRNGNN64G46H005D','9.0'),

('7134af62-a1e2-43df-a150-1ec98e44c8fb', '850216a6-3c23-4aca-ad7d-ee0b1916bc7b','2018-05-21 15:32:25+01',NULL,'12.50','BRNGNN64G46H005D','7.5');

--This insertion attempt fails since the specified hourly wage is different from the one declared in the employee table (see trigger CheckWage)

INSERT INTO TimeSlot(TimeSlotID, TaskID, CurTime, Notes, HourlyWage, FiscalCode, Hours) VALUES

('b4614a62-215a-4cbe-a080-4ac35032aeef','353bf9d0-183e-469d-978a-9484f6c25b15','2018-08-02 04:32:25+01',NULL,'6.55','LRNBTT96C01D149A','2.0');

--Expense

INSERT INTO Expense(Type) VALUES

('TripExpenses'),

('FuelExpenses'),

('DesignMaterials');

--Has

INSERT INTO Has(TimeSlotID, Type, Cost, Description) VALUES

('69aed574-6572-42f0-863e-c7ba2260d752','DesignMaterials','250.80','Paper and some other materials needed for the design'),

('12476d9b-058c-4a71-b33b-bdca507e53c6','TripExpenses','540.12',NULL),

('31f1de04-e117-4463-8eef-ba0598502e66','FuelExpenses','320.00','Fuel for 150 km');

Java code for inserting and retrieving the PDFs from the database:

Principal Queries

In this section, we report four queries to navigate the database:

1. Retrieve general information about the customers and the projects that they have requested;
2. Retrieve the number of task that each department is in charge of;
3. Retrieve how many hours the employees have spent on the projects;
4. Retrieve the structure of the tasks inside the projects;

1.

-- Retrieve customer's names, surnames and the names of the projects assigned by them

SELECT surname, name, title

FROM Contact AS Co INNER JOIN Customer AS Cu ON Co.FiscalCode = Cu.FiscalCode

INNER JOIN Request AS R ON Cu.FiscalCode = R.FiscalCode

INNER JOIN Project as P ON R.ProjectID = P.ProjectID

ORDER BY surname, name, title ASC;

Immagine che contiene screenshot

Descrizione generata automaticamente

*Figure 2 Results for Query 1*

2.

-- Retrieve the names of the departments and the number of tasks to which they are assigned

SELECT d.Name, COUNT(TaskID) as number\_of\_tasks

FROM Department as d

LEFT OUTER JOIN TASK AS t ON d.Name = t.Name

GROUP BY d.Name

ORDER BY d.Name ASC;

Immagine che contiene screenshot

Descrizione generata automaticamente

*Figure 3 Results for Query 2*

3.

-- Return how many hours the employees have spent on the projects

SELECT Title, Surname, Name, SUM(employeeHours.Hours) AS Total\_Hours

FROM (

SELECT Contact.Surname, Contact.Name, Task.TaskID, TimeSlot.Hours

FROM Employee

INNER JOIN TimeSlot ON Employee.FiscalCode = TimeSlot.FiscalCode

INNER JOIN Task ON TimeSlot.TaskID = Task.TaskID

INNER JOIN Contact ON TimeSlot.FiscalCode = Contact.FiscalCode

) AS employeeHours

INNER JOIN Compose ON employeeHours.TaskID = Compose.Child

INNER JOIN Project ON Compose.ProjectID = Project.ProjectID

GROUP BY Title, Surname, Name

ORDER BY Title, Surname, Name ASC;

4.

--Display the structure of the projects along with their descriptions

WITH RECURSIVE proj\_subpart AS (

SELECT Parent, Child, ProjectID, 1 AS Depth

FROM Task as t INNER JOIN Compose AS C

ON t.taskID = C.Child

WHERE isRoot = TRUE

UNION ALL

SELECT C.Parent, C.Child, C.ProjectID, Depth + 1

FROM proj\_subpart AS pr INNER JOIN Compose AS C ON C.Parent = pr.Child

)

SELECT p.Title AS Project\_Name, tmt.TemplateID AS Task\_Description, Depth, tmt.Description AS General\_task\_description

FROM Project AS p INNER JOIN proj\_subpart AS ps ON p.ProjectID = ps.ProjectID

INNER JOIN Task AS t ON ps.Child = t.taskID

INNER JOIN Template AS tmt ON t.TemplateID = tmt.TemplateID

ORDER BY p.Title, Depth ASC;

Immagine che contiene testo

Descrizione generata automaticamente

*Figure 5 Results for Query 4*

Stored Procedure

The following stored procedure is required due to the high demand of the employee’s report of the hours.

-- Connect to ennedue db

\c ennedue

-- Stored procedure to select all the timeslots of an employee in a certain period

-- Replace to modify an existing version of this function

CREATE OR REPLACE FUNCTION viewEmployeeHours(targetEmployeeFiscalCode VARCHAR(16), fromDate DATE, toDate DATE)

RETURNS TABLE(

TimeSlotId UUID,

Hours FLOAT(2)

)

LANGUAGE plpgsql

AS $$

DECLARE

-- The fiscal code of the employee

fiscalCodeCheck VARCHAR;

BEGIN

-- Check whether the employee fiscal code exists

SELECT FiscalCode INTO fiscalCodeCheck

FROM Employee WHERE FiscalCode = targetEmployeeFiscalCode;

IF NOT FOUND THEN

RAISE EXCEPTION 'employee fiscal code % does not exist', targetEmployeeFiscalCode;

ELSE

IF fromDate > toDate THEN

RAISE EXCEPTION 'date interval is not valid';

ELSE

-- Return all the timeslots of the employee

RETURN QUERY

SELECT TimeSlot.TimeSlotID, TimeSlot.Hours

FROM Employee

INNER JOIN TimeSlot ON Employee.FiscalCode = TimeSlot.FiscalCode

INNER JOIN Task ON TimeSlot.TaskID = Task.TaskID

WHERE TimeSlot.CurTime::DATE >= fromDate

AND TimeSlot.CurTime::DATE < toDate

AND Employee.FiscalCode = targetEmployeeFiscalCode;

END IF;

END IF;

END;

$$;

JDBC Implementations of the Principal Queries and Visualization

Hereafter, we report a java class which read the data from the database and print the results on screen.

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

\* This method prints all the Tasks associated to a Job

\*

\* @author Winniest Team

\* @version 1.00

\*/

public class PrintProjects {

/\*\*

\* 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://localhost/ennedue";

/\*\*

\* The username for accessing the database

\*/

private static final String USER = "postgres";

/\*\*

\* The SQL statement to be executed

\*/

private static final String SQL = "SELECT name, surname, title, StartDate, EndDate, Location, Deadline, EstimatedHours, HoursSpent FROM Customer AS Cu INNER JOIN Contact AS Co ON Cu.FiscalCode = Co.FiscalCode INNER JOIN Request AS R ON Cu.FiscalCode = R.FiscalCode INNER JOIN Project as P ON R.ProjectID = P.ProjectID;";

/\*\*

\* The password for accessing the database, initialized during runtime

\*/

private static String password;

/\*\*

\* List all the projects in the database with the customer that as demanded it

\*

\* @param args

\* command-line arguments used for inserting the password.

\*/

public static void main(String[] args) {

// Retrive the password from the input

if (args.length < 1){

// Exit the program with a generic error

System.out.printf("ERROR: Insert a password in the input.%n");

System.exit(-1);

}

// Get the password from the input

password = args[0];

// The connection to the DBMS

Connection con = null;

// The statement to be executed

Statement stmt = null;

// The results of the statement execution

ResultSet result = null;

// Start time of a statement

long start;

// End time of a statement

long end;

// Variables for saving the result from the database

// Data about the project

String name = null;

String surname = null;

String title = null;

String startDate = null;

String endDate = null;

String location = null;

String deadline = null;

int estimatedHours = -1;

double hoursSpent = -1.0;

// Register the JDBC driver

try {

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());

// Exit with an error code

System.exit(-1);

}

// Connect to the database

try {

// Count the necessary time for the conection

start = System.currentTimeMillis();

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

// Stop timing

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 and count the time

start = System.currentTimeMillis();

stmt = con.createStatement();

// Stop timing

end = System.currentTimeMillis();

System.out.printf(

"Statement successfully created in %,d milliseconds.%n",

end-start);

// Execute the query

start = System.currentTimeMillis();

result = stmt.executeQuery(SQL);

end = System.currentTimeMillis();

System.out.printf("Query: %n%s%nsuccessfully executed in %,d milliseconds.%n",

SQL, end - start);

System.out.printf("%nQUERY RESULT:%n");

// Print all the result of the query

while (result.next()) {

// Read the customer's data

name = result.getString("name");

surname = result.getString("surname");

// Read the project's data

title = result.getString("title");

startDate = result.getString("startDate");

endDate = result.getString("endDate");

location = result.getString("location");

deadline = result.getString("deadline");

estimatedHours = result.getInt("estimatedHours");

hoursSpent = result.getDouble("hoursSpent");

// Print result

// EndDate can be null if the project is still in progress so we can have two different kind of responses

if (endDate == null){

// The project is still in progress

System.out.printf("%nCustomer: %s %s %nTitle: %s, %s %nstarted on %s and still in progress%ndue on %s%nStatistics: %.2f / %d%n%n",

name, surname, title, location, startDate, deadline, hoursSpent, estimatedHours);

} else {

// The project is completed

System.out.printf("%nCustomer: %s %s %nTitle: %s, %s %nstarted on %s and ended on %s%ndue on %s%nStatistics: %.2f / %d%n%n",

name, surname, title, location, startDate, endDate, deadline, hoursSpent, estimatedHours);

}

}

} catch (SQLException e) {

System.out.printf("ERROR: Database access error%n");

printErrorMessages(e);

} finally {

// Close all the connection in the reverse order of their creation

try {

// Close the used resources

if (result != null) {

// Count time

start = System.currentTimeMillis();

result.close();

// Stop timing

end = System.currentTimeMillis();

System.out.printf("Result set successfully closed in %,d milliseconds.%n",

end-start);

}

if (stmt != null) {

// Count time

start = System.currentTimeMillis();

stmt.close();

// Stop timing

end = System.currentTimeMillis();

System.out.printf("Statement successfully closed in %,d milliseconds.%n",

end-start);

}

if (con != null) {

// Count time

start = System.currentTimeMillis();

con.close();

// Stop timing

end = System.currentTimeMillis();

System.out.printf("Connection successfully closed in %,d milliseconds.%n",

end-start);

}

System.out.printf("All the resources successfully released.%n");

} catch (SQLException e) {

System.out.printf("Error while releasing resources:%n");

printErrorMessages(e);

} finally {

// Release resources to the garbage collector

result = null;

stmt = null;

con = null;

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

}

}

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

}

private static void printErrorMessages(SQLException e){

// Get all the errors

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();

}

}

}