UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Foundations of Databases A.Y. 2021-2022

Homework 3 – Physical Design

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: December 17, 2021

Team acronym	TAGMS	
Last Name	**First Name**	**Student Number**
Giuliani	Amedeo	2005797
Esposito	Vittorio	2005795
Basso	Marco	2005796
Zanini	Samuele	2019038

Variations to the Relational Schema

From the previous homework we have found some typos within the Data Dictionary section. The respective corrections are shown below: - In Employee the "Registration_date_ID" attribute has been renamed to "Registration_date". - In Draws_from, the description of the "Lot_ID" attribute is changed from "Foreign Key to Lot, Primary key to Lot" to "Foreign Key to Lot, Primary key" - In Ship, the "Employee_ID" attribute is not a serial but a text type. In particular, a new domain called "taxnumber" has been defined, to which the Employee entity ID is associated.

Physical Schema

[Report SQL statements for the database creation]

– Database Creation

```
-- Database Creation
CREATE DATABASE tagmsdb OWNER admin ENCODING = 'UTF8';

-- Create new Schema (you can use the public schema without creating a new one, if you want)
DROP SCHEMA IF EXISTS Tagms CASCADE;
CREATE SCHEMA Tagms;

-- Create new domains
CREATE DOMAIN Tagms.emailaddress AS VARCHAR(254)
CONSTRAINT properemail CHECK (((VALUE)::text ~* '^[A-Za-z0-9._%]+@[A-Za-z0-9.]+[.][A-Za-z]+$'::text));
CREATE DOMAIN Tagms.phonenumber AS VARCHAR(17)
CONSTRAINT properphonenumber CHECK ( VALUE ~ ^(\((00|\+)39\)|(00|\+)39)?(38[890]|34[7-90]|36[680]|33[3-90]|3
CREATE DOMAIN Tagms.vatnumber AS CHAR(11)
CONSTRAINT properVAT CHECK (((VALUE)::text ~* ^[0-9]*$::text));
CREATE DOMAIN Tagms.taxnumber AS CHAR(16)
CONSTRAINT taxnumber CHECK (VALUE ~ ^[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]$ );
CREATE DOMAIN Tagms.price AS MONEY
CONSTRAINT price CHECK (VALUE > 0);
CREATE DOMAIN Tagms.quantity AS INTEGER
CONSTRAINT quantity CHECK (VALUE >= 1);
CREATE DOMAIN Tagms.dimension AS INTEGER
CONSTRAINT dimension CHECK (VALUE > 0);
CREATE DOMAIN Tagms.percentage AS DECIMAL(4,1)
CONSTRAINT percentage CHECK (VALUE >= 0 and VALUE =< 100);

-- Table Creation (create tables in the correct order, i.e. use FKeys only referring to
already created tables)

CREATE TABLE Tagms.product (
product_id SERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500),
production_cost Tagms.price NOT NULL,
price_increase Tagms.price NOT NULL,    %TODO: price_increase Numeric(5,2) or Double precision or Range or T
volume Tagms.dimension NOT NULL,
net_weight Tagms.dimension NOT NULL,
package_weight Tagms.dimension NOT NULL,
nutritional_facts VARCHAR(500) NOT NULL,
product_category_id SMALLINT NOT NULL,
```

```
FOREIGN KEY (product_category_id) REFERENCES Tagms.product_category(product_category_id)
);

CREATE TABLE Tagms.product_category (
product_category_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.item (
item_id SERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500),
quantity Tagms.quantity NOT NULL,
minimum_quantity Tagms.quantity NOT NULL,
item_category_id SMALLINT NOT NULL,
FOREIGN KEY (item_category_id) REFERENCES Tagms.item_category(item_category_id)
);

CREATE TABLE Tagms.item_category (
item_category_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.specify (
item_id INTEGER,
contract_id INTEGER,
price Tagms.price NOT NULL,
purchased_quantity Tagms.quantity NOT NULL,
FOREIGN KEY (item_id) REFERENCES Tagms.item(item_id),
FOREIGN KEY (contract_id) REFERENCES Tagms.contract(contract_id),
PRIMARY KEY (item_id, contract_id)
);

CREATE TABLE Tagms.contract (
contract_id SERIAL PRIMARY KEY,
description VARCHAR(500),
contract_date TIMESTAMP WITH TIME ZONE NOT NULL,
delivery_date TIMESTAMP WITH TIME ZONE NOT NULL,
expiration_date TIMESTAMP WITH TIME ZONE NOT NULL,
supplier_id Tagms.vatnumber NOT NULL,
employee_id Tagms.taxnumber NOT NULL,
FOREIGN KEY (supplier_id) REFERENCES Tagms.supplier(vat_number),
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(tax_number)
);

CREATE TABLE Tagms.package (
package_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500),
weight Tagms.dimension NOT NULL,
height Tagms.dimension NOT NULL,
width Tagms.dimension NOT NULL,
```

```
depth Tagms.dimension NOT NULL,
package_category_id SMALLINT NOT NULL,
FOREIGN KEY (package_category_id) REFERENCES Tagms.package_category(package_category_id)
);

CREATE TABLE Tagms.package_category (
package_category_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);


CREATE TABLE Tagms.lot (
lot_id SERIAL PRIMARY KEY,
expiration_date TIMESTAMP WITH TIME ZONE NOT NULL,
product_id INTEGER NOT NULL,
product_quantity Tagms.quantity NOT NULL,
package_id INTEGER NOT NULL,
package_quantity Tagms.quantity NOT NULL,
lot_discount Tagms.percentage  NOT NULL,
vat Tagms.percentage NOT NULL,
lot_price Tagms.price NOT NULL,
FOREIGN KEY (package_id) REFERENCES Tagms.package(package_id),
FOREIGN KEY (product_id) REFERENCES Tagms.product(product_id)
);


CREATE TABLE Tagms.order (
order_id SERIAL PRIMARY KEY,
net_price Tagms.price NOT NULL,
taxes Tagms.price NOT NULL,
order_date TIMESTAMP WITH TIME ZONE NOT NULL,
employee_id Tagms.taxnumber NOT NULL,
customer_id Tagms.vatnumber NOT NULL,
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(employee_id),
FOREIGN KEY (customer_id) REFERENCES Tagms.customer(customer_id)
);

CREATE TABLE Tagms.customer (
vat_number Tagms.vatnumber PRIMARY KEY,
customer_name VARCHAR(50) NOT NULL,
phone_number Tagms.phonenumber NOT NULL,
address VARCHAR(100) NOT NULL,
email_address Tagms.emailaddress NOT NULL,
registration_date TIMESTAMP WITH TIME ZONE NOT NULL
);

CREATE TABLE Tagms.draws_from (
lot_id INTEGER PRIMARY KEY,
order_id INTEGER NOT NULL,
FOREIGN KEY (lot_id) REFERENCES Tagms.lot(lot_id),
FOREIGN KEY (order_id) REFERENCES Tagms.lot(order_id),
);
```

```
CREATE TABLE Tagms.made_up_of_1 (
product_id INTEGER,
item_id INTEGER,
quantity Tagms.quantity NOT NULL,
FOREIGN KEY (product_id) REFERENCES Tagms.product(product_id),
FOREIGN KEY (item_id) REFERENCES Tagms.item(item_id),
PRIMARY KEY (product_id, item_id)
);

CREATE TABLE Tagms.made_up_of_2 (
package_id INTEGER,
item_id INTEGER,
quantity Tagms.quantity NOT NULL,
FOREIGN KEY (package_id) REFERENCES Tagms.package(package_id),
FOREIGN KEY (item_id) REFERENCES Tagms.item(item_id),
PRIMARY KEY (package_id, item_id)
);

CREATE TABLE Tagms.ship (
order_id INTEGER PRIMARY KEY,
employee_id Tagms.taxnumber NOT NULL,
shipping_date TIMESTAMP WITH TIME ZONE NOT NULL,
track_num NUMERIC(10),
FOREIGN KEY (order_id) REFERENCES Tagms.order(order_id),
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(employee_id),
);

CREATE TABLE Tagms.employee (
tax_number Tagms.taxnumber PRIMARY KEY,
first_name VARCHAR(30) NOT NULL,
last_name VARCHAR(30) NOT NULL,
phone_number Tagms.phonenumber NOT NULL,
email_address Tagms.emailaddress NOT NULL,
birth_date DATE,
hiring_date DATE NOT NULL,
still_working BOOLEAN NOT NULL,
role_id SMALLINT NOT NULL,
FOREIGN KEY (role_id) REFERENCES Tagms.role(role_id)
);

CREATE TABLE Tagms.role (
role_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.work (
department_id SMALLINT,
employee_id Tagms.taxnumber,
FOREIGN KEY (department_id) REFERENCES Tagms.department(department_id)
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(employee_id)
PRIMARY KEY (department_id, employee_id)
);
```

```
CREATE TABLE Tagms.department (
department_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.supplier (
vat_number Tagms.vatnumber PRIMARY KEY
supplier_name VARCHAR(50) NOT NULL,
phone_number Tagms.phonenumber NOT NULL,
email_address Tagms.emailaddress NOT NULL,
address VARCHAR(100) NOT NULL,
registration_date DATE NOT NULL
);
```

Populate the Database: Example

[Report SQL statements for the database population: one row per table is enough for the HW, but we suggest to insert a sufficient amount of data in your DB to run queries]

```
-- Employee
INSERT INTO Tagms.employee(TAX_numeber, First_name, Last_name, Phone_number, Email_address,
Birth_date, Hiring_date, Still_working, Role_ID) VALUES
('','','','','','','','','');

-- Work
INSERT INTO Tagms.work(Department_ID, Employee_ID) VALUES
('','');

-- Departament
INSERT INTO Tagms.departament(Department_ID, Name,Description) VALUES ('','','');

-- Role
INSERT INTO Tagms.role(Role_ID, Name,Description) VALUES
('','','');

-- Ship
INSERT INTO Tagms.ship(Order_ID, Employee_ID, Shipping_date,Track_num) VALUES
('','','','');

-- Supplier
INSERT INTO Tagms.supplier(VAT_numeber, Supplier_name, Phone_number, Address, Email_address,
Registration_date) VALUES
('','','','','','');

-- Contract
INSERT INTO Tagms.contract(Contract_ID, Description, Contract_date, Delivery_date,
Expiration_date, Supplier_ID, Employee_ID) VALUES
('','','','','','','');
```

```
-- Item
INSERT INTO Tagms.item(Item_ID, Name, Description,
Quantity, Item_Category_ID, Minimum_quantity) VALUES
('','','','','','');

-- Item_Category
INSERT INTO Tagms.item_Category(Item_Category_ID, Name, Description) VALUES
('','','');

-- Specify
INSERT INTO Tagms.specify(Item_ID, Contract_ID, Purchase_Quantity, Price) VALUES
('','','','');


-- Product
INSERT INTO Tagms.product(Product_ID, Name, Description, Product_cost, Price_increase,
Volume, Net_weight, Nutritional_Facts, Product_Category_ID)  VALUES
('', '', '', '', '', '', '', '', '');

-- Product_Category
INSERT INTO Tagms.product_Category(Product_Category_ID, Name, Description) VALUES
('','','');

-- Made_Up_Of_1
INSERT INTO Tagms.Made_Up_Of(Product_ID, Item_ID,Quantity) VALUES
('','','');


-- Package
INSERT INTO Tagms.package(Package_ID, Name, Description, Weight, Height, Width, Depth,
Package_Category_ID)VALUES
('', '', '', '', '', '', '', '');

-- Package_Category
INSERT INTO Tagms.product_Category(Package_Category_ID, Name, Description) VALUES
('','','');

-- Made_Up_Of_2
INSERT INTO Tagms.Made_Up_Of(Package_ID, Item_ID,Quantity) VALUES
('','','');



-- Order
INSERT INTO Tagms.order(Order_ID,Net_price, Taxes, Order_date, Order_paid, Employee_ID,
Customer_ID) VALUES
('','', '', '', '', '', '')

-- Customer
INSERT INTO Tagms.customer(VAT_numeber, Customer_name, Registration_date, Address,
Email_address, Phone_number) VALUES
('', '', '', '','', '')
```

```
-- Lot
INSERT INTO Tagms.lot(Lot_ID, Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
Package_Quantity,Package_ID, Product_ID) VALUES
('', '','','', '', '', '','', '');

-- Draws_from
INSERT INTO Tagms.Made_Up_Of(Lot_ID, Order_ID) VALUES
('','');
```

Principal Queries

[Write some of the queries to be performed to satisfy your functional requirements. 3-4 queries are enough, try to use the techniques seen at lecture (aggregate functions, group by, subqueries,...)]

```
SELECT t1.attr1
FROM table1 as t1
    LEFT JOIN table2 as t2 ON t1.attr2=t2.attr1
WHERE t1.attr3=1
```

JDBC Implementations of the Principal Queries and Visualization

[Report java code to test your DB. Perform 1-2 queries and display the results]

```
/*
 *  Write here your java code
 */

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Group Members Contribution