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
Insert last name here	Insert first name here	Insert student number here
Zanini	Samuele	2019038

Variations to the Relational Schema

[Describe here variations and/or corrections to the relational schema of the previous homework, if present. Otherwise, report only the relational schema.]

Physical Schema

[Report SQL statements for the 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.vatnumber AS VARCHAR(15)
CONSTRAINT properVAT CHECK (((VALUE)::text ~* '[A-Za-z0-9]{5,}'::text));
CREATE DOMAIN Tagms.serial AS CHAR(10)
CONSTRAINT properserial CHECK (((VALUE)::text ~* '[0-9]{10}'::text));
CREATE DOMAIN Tagms.phonenumber AS VARCHAR(17)
CONSTRINT properphonenumber CHECK ( VALUE ~ ^(\((00|\+)39\)|(00|\+)39)?(38[890]|34[7-90]|36[680]|33[3-90]|32

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

CREATE TABLE Tagms.product_category (
product_category_id SERIAL PRIMARY KEY,
name VARCHAR(32) NOT NULL,
description VARCHAR(500),
);

CREATE TABLE Tagms.package_category (
package_category_id SERIAL PRIMARY KEY,
name VARCHAR(32) NOT NULL,
description VARCHAR(500),
);

CREATE TABLE Tagms.item_category (
item_category_id SERIAL PRIMARY KEY,
name VARCHAR(32) NOT NULL,
description VARCHAR(500),
);

CREATE TABLE Tagms.product (
product_id SERIAL PRIMARY KEY,
name VARCHAR(32) NOT NULL,
description VARCHAR(500),
production_cost DOUBLE PRECISION NOT NULL,
price_increase DOUBLE PRECISION NOT NULL,
volume INTEGER NOT NULL,
```

```
net_weight INTEGER NOT NULL,
package_weight INTEGER NOT NULL,
nutritional_facts VARCHAR(500) NOT NULL,
product_category_id INTEGER NOT NULL,
FOREIGN KEY (product_category_id) REFERENCES Tagms.product_category(product_category_id),
);

CREATE TABLE Tagms.package (
package_id SERIAL PRIMARY KEY,
name VARCHAR(32) NOT NULL,
description VARCHAR(500),
weight INTEGER NOT NULL,
height INTEGER NOT NULL,
width INTEGER NOT NULL,
depth INTEGER NOT NULL,
package_category_id INTEGER NOT NULL,
FOREIGN KEY (package_category_id) REFERENCES Tagms.package_category(package_category_id),
);

CREATE TABLE Tagms.item (
item_id SERIAL PRIMARY KEY,
name VARCHAR(32) NOT NULL,
description VARCHAR(500),
quantity INTEGER NOT NULL,
minimum_quantity INTEGER NOT NULL,
item_category_id INTEGER NOT NULL,
FOREIGN KEY (item_category_id) REFERENCES Tagms.item_category(item_category_id),
);

CREATE TABLE Tagms.made_up_of_1 (
product_id INTEGER,
item_id INTEGER,
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,
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.role (
role_id TINYINT SERIAL PRIMARY KEY,
name varchar(30) NOT NULL,
description varchar(256)
);

CREATE TABLE Tagms.employee (
tax_number VARCHAR(17) PRIMARY KEY,
first_name VARCHAR(20),
```

```
last_name VARCHAR(20),
phone_number phonenumber,
email_address emailaddress,
birth_date DATE,
hiring_date DATE,
still_working BOOL,
role_id TINYINT,
FOREIGN KEY (role_id) REFERENCES Tagms.role(role_id)
);
```

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