Wyższa Szkoła Ekonomii
i Informatyki w Krakowie

Projekt zaliczeniowy z przedmiotu „Wprowadzenie do baz danych”

Projektowanie systemu segregacji zamówień oraz produktów dla hurtowni części samochodowych.

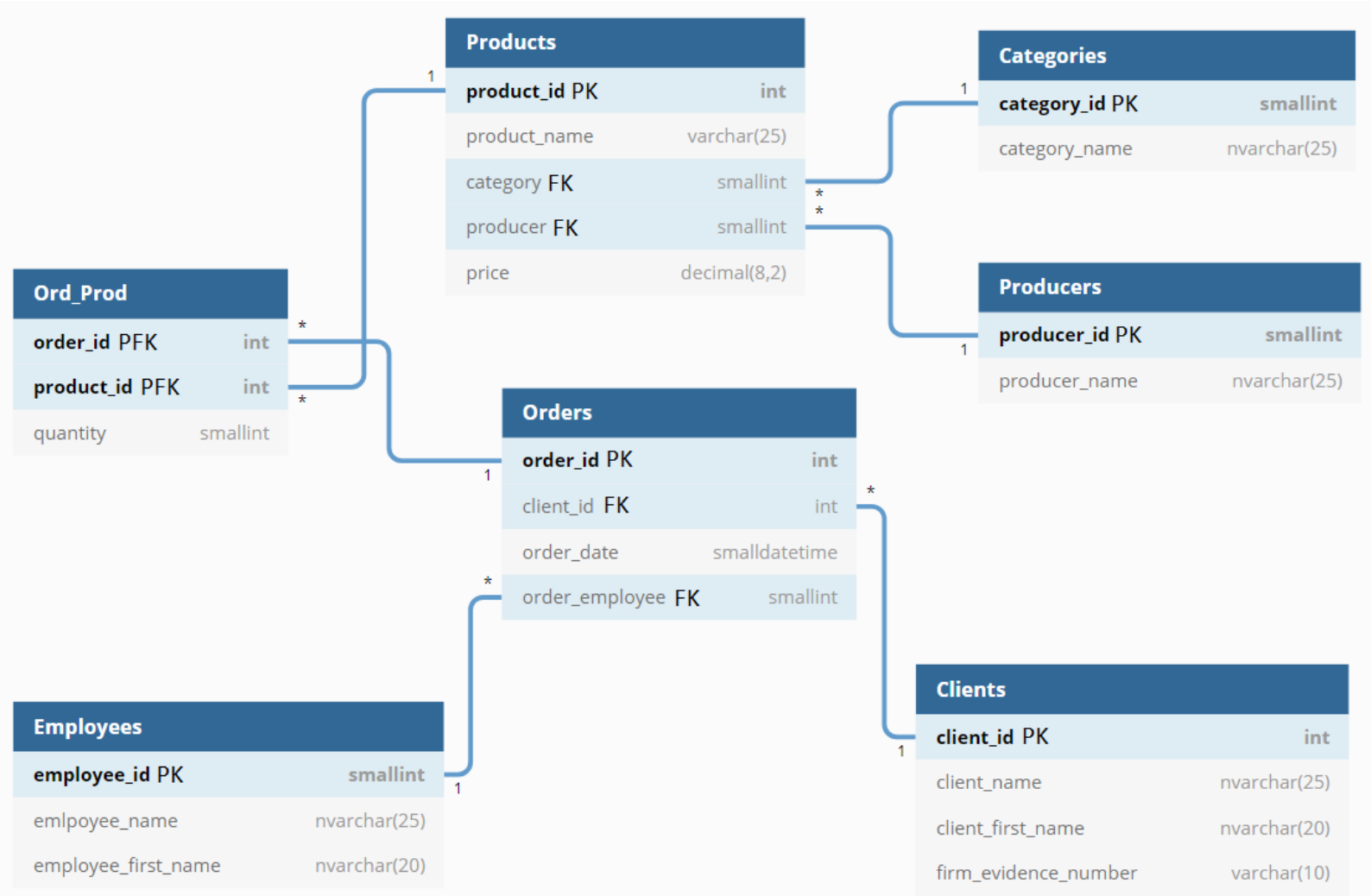
Damian Jarosz

Informatyka i ekonometria 2019/2020

Grupa lab2/1/IES, nr albumu 12573

Dyrektor hurtowni z częściami samochodowymi zauważył, że obroty jego hurtowni spadły. Mimo tego, że mnóstwo klientów odwiedza hurtownie oraz kompletuje zamówienia, nie finalizuje ich. W skrzynce na opinie pojawiają się anonimowe listy niezadowolonych klientów narzekających na nieprzyjazną, ociągającą się obsługę, która „sprawia wrażenie, jakby siedziała w pracy na siłę”. Ponadto pracownicy tracą mnóstwo czasu na odszukanie nieskategoryzowanych produktów, co przekłada się na wzrost czasu oczekiwania klienta na realizację zamówienia oraz kumulowanie się klientów oczekujących w kolejce.

Dyrektor wraz z zarządem ustalili, że potrzebna jest restrukturyzacja procesu zarządzania hurtownią. W tym celu chcą stworzyć bazę danych przechowującą zamówienia, wraz z odpowiadającymi im klientami oraz skategoryzować wszystkie produkty dostępne w hurtowni. Głównym wymogiem firmy jest to, by pracownicy nie mogli niechlujnie wpisywać danych do systemu, jak to miało miejsce do tej pory (nieuporządkowane dane ilościowe produktów powodowały opóźnienia w realizacji zamówień).



Struktura bazy danych:

Tabele:

1. **Categories:**
   - category_id – smallint, pole klucza głównego, autonumerowane, typ danych typ smallint został użyty, ponieważ 32767 jest odpowiednią liczbą, która bez problemu pomieści żądaną ilość kategorii
   - category_name – nvarchar(25), UNIQUE – użyta klauzula UNIQUE uniemożliwia dublowanie kategorii (np. 2x hamulce);
2. **Producers:**
   - producer_id – smallint, pole klucza głównego, autonumerowane, typ danych smallint został użyty, ponieważ 32767 jest odpowiednią liczbą, która bez problemu pomieści żądaną ilość producentów
   - producer_name – nvarchar(25) – w przeciwieństwie do categories.category_name pole nie zostało obłożone klauzulą UNIQUE, ponieważ po bankructwie jednego przedsiębiorstwa można założyć inne o takiej samej nazwie, uniemożliwienie wpisania producentów o takich samych nazwach w przyszłości mogło by powodować komplikacje i niespójności informacji;
3. **Products:**
   - product_id – int, pole klucza głównego, autonumerowane
   - product_name varchar(25) – 25 znaków to odpowiednia ilość do opisania nazwy produktu
   - category – smallint, połączone FK z categories.category_id
   - producer – smallint, połączone FK z producers.producer_id
   - price – decimal(8,2) – 999999,99 jest odpowiednią maksymalną liczbą do określenia ceny części samochodowych, nie ma potrzeby jej zwiększania i zabierania większej ilości miejsca na dysku;
4. **Clients:**
   - client_id – int, pole klucza głównego, autonumerowane – nie ograniczamy dostępnej ilości klientów w bazie mniejszym typem danych
   -client_name – nvarchar(25)
   -client_first_name – nvarchar(20)
   -firm_evidence_number – varchar(10) – opcjonalne pole na numer NIP dla klientów biznesowych (w Polsce 10 cyfr, za granicą 8-10 cyfr lub znaków);
5. **Employees:**
   - employee_id – smallint, pole klucza głównego, autonumerowane, w przeciwieństwie do clients.client_id typ smallint jest odpowiedni, ponieważ firma nie planuje aż tak dużej rotacji pracowników, by ten typ danych miał być niewystarczający
   - employee_name – nvarchar(25)
   - employee_first_name – nvarchar(20);

6. **Orders:**
   - order_id – int, pole klucza głównego, autonumerowane – nie ograniczamy ilości potencjalnych zamówień mniejszym typem danych
   - client_id – int, połączone FK z clients.client_id
   - order_date – typ smalldatetime jest odpowiedni do jednoznacznego opisania daty i czasu złożenia zamówienia bez niepotrzebnego używania większego typu danych, nałożona klauzula CHECK mająca sprawdzać, czy zamówienie nie jest składane z datą przyszłą, może zostać wpisana aktualna data, lecz nie wcześniejsza niż 2016-02-03(jest to data założenia firmy)
   - order_employee – smallint – pole połączone FK z employees.employee_id
7. **Ord_Prod:**
   - Tabela kojarząca zamówienia i zawarte w nich produkty oraz ich ilość, wielopolowy klucz główny na polach order_id oraz product_id uniemożliwia zdublowanie danego produktu w danym zamówieniu

Relacje:

1. **PK categories.category_id > FK products.category** – produkt może być przypisany tylko do jednej kategorii, lecz w danej kategorii może się znajdować wiele produktów
2. **PK producers.producer_id > FK products.producer –** produkt może mieć tylko jednego producenta, lecz producent może produkować wiele podzespołów
3. **PFK dla pól order_id oraz product_id w tabeli Ord_Prod > PK dla pól products.product_id oraz orders.order_id –** tabela Ord_Prod kojarzy produkty z zamówieniami, uniemożliwia skojarzenie jednego produktu więcej niż raz z danym zamówieniem
4. **PK clients.client_id > FK orders.client_id –** klient może złożyć wiele zamówień, lecz dane zamówienie może mieć przypisanego tylko jednego klienta
5. **PK employees.employee_id – FK orders.order_employee –** do danego zamówienia może być przypisany tylko jeden pracownik, lecz każdy pracownik może obsługiwać wiele zleceń

Kod bazy danych, poniżej znajdują się przykładowe inserty oraz opisane zapytania:

```
CREATE DATABASE Mydb
go
USE Mydb
go
CREATE TABLE Categories(
category_id smallint IDENTITY(1,1) PRIMARY KEY,
category_name nvarchar(25) UNIQUE
);
CREATE TABLE Producers(
producer_id smallint IDENTITY(1,1) PRIMARY KEY,
producer_name nvarchar(25)
);
CREATE TABLE Products(
product_id int IDENTITY(1,1) PRIMARY KEY,
product_name varchar(25) NOT NULL,
category smallint NOT NULL,
producer smallint NOT NULL,
price decimal(8,2) NOT NULL,
CONSTRAINT c5 FOREIGN KEY(category)
REFERENCES Categories(category_id)
ON DELETE no action
ON UPDATE cascade,
CONSTRAINT c6 FOREIGN KEY(producer)
REFERENCES Producers(producer_id)
ON DELETE no action
ON UPDATE cascade,
);
CREATE TABLE Clients(
client_id int IDENTITY(1,1) PRIMARY KEY,
client_name nvarchar(25) NOT NULL,
client_first_name nvarchar(20) NOT NULL,
firm_evidence_number varchar(10)
);
CREATE TABLE Employees(
employee_id smallint IDENTITY(1,1) PRIMARY KEY,
employee_name nvarchar(25) NOT NULL,
employee_first_name nvarchar(20) NOT NULL
);
CREATE TABLE Orders(
order_id int IDENTITY(1,1) PRIMARY KEY,
client_id int NOT NULL,
order_date smalldatetime NOT NULL CHECK(order_date <= GETDATE()
AND order_date >='2016-02-03'),
order_employee smallint NOT NULL
CONSTRAINT c1 FOREIGN KEY(client_id)
REFERENCES Clients(client_id)
ON DELETE no action
ON UPDATE cascade,
CONSTRAINT c2 FOREIGN KEY(order_employee)
REFERENCES Employees(employee_id)
ON DELETE no action
ON UPDATE cascade,
);
CREATE TABLE Ord_Prod(
order_id int NOT NULL,
product_id int NOT NULL,
```

```
quantity smallint NOT NULL,
PRIMARY KEY (order_id, product_id),
CONSTRAINT c3 FOREIGN KEY(order_id)
REFERENCES Orders(order_id)
ON DELETE no action
ON UPDATE cascade,
CONSTRAINT c4 FOREIGN KEY(product_id)
REFERENCES Products(product_id)
ON DELETE no action
ON UPDATE cascade,
);
INSERT INTO Categories VALUES('Absorbers'),
('Tyres'),
('Brakes'),
('Bulbs'),
('Filters');

INSERT INTO Producers VALUES('Bosch'),
('Exide'),
('Castrol'),
('Denso'),
('Filtron');

INSERT INTO Products VALUES ('Hamilton', 3, 1, 250.5);
INSERT INTO Products VALUES ('Frigo II', 2, 3, 170.99);
INSERT INTO Products VALUES ('B.I.G.', 1, 4, 570.00);
INSERT INTO Products VALUES ('h7', 4, 4, 12.00);
INSERT INTO Products VALUES ('Air filter 99x2', 5, 5, 56.99);

INSERT INTO Clients VALUES('Kowalski', 'Jan', 1234567890);
INSERT INTO Clients(client_name, client_first_name) VALUES('Nowak', 'Jan');
INSERT INTO Clients VALUES('Dadacki', 'Kazimierz', 5556662213);

INSERT INTO Employees VALUES('Aacki', 'Jan'),
('Babacki', 'Stefan'),
('Cacacki', 'Zbigniew'),
('Dadacki', 'Olgierd');

INSERT INTO Orders VALUES (1,'2018-03-06 16:15',1);
INSERT INTO Orders VALUES (2,'2019-04-05 10:20',2);
INSERT INTO Orders VALUES (3,'2018-02-01 12:00',3);

INSERT INTO Ord_Prod VALUES(1,2,2),
(1,1,3),
(1,4,2),
(2,5,1),
(2,3,4),
(2,4,2),
(3,2,1),
(3,1,1),
(3,5,1);


SELECT product_id, product_name, category_name, producer_name, price
       FROM products p left outer join categories c on p.category=c.category_id
                       left outer join producers pr on p.product_id=pr.producer_id
       WHERE producer_name = 'Bosch'

/* Zapytanie pokazujące id, nazwę, kategorię, producenta oraz cenę wszystkich
produktów firmy Bosch */
```

```sql
SELECT order_id, order_date, employee_name
FROM Orders o left outer join Employees e ON o.order_employee=e.employee_id

/* Zapytanie pokazujące wszystkie zamówienia, datę ich powstania oraz przypisanych do
nich pracowników */

SELECT order_id, order_date, employee_name
FROM Orders o left outer join Employees e ON o.order_employee=e.employee_id WHERE
employee_name='Aacki'

/* Zapytanie pokazujące wszystkie zamówienia, do których przypisany jest dany
pracownik */

SELECT o.order_id, o.order_date, c.client_first_name, c.client_name, p.product_name,
ct.category_name, p.price, op.quantity, (p.price*op.quantity) AS total_value,
employee_name AS assigned_employee
FROM Orders o left outer join employees e ON o.order_employee=e.employee_id
                      left outer join Ord_Prod op ON o.order_id=op.order_id
                      left outer join Products p ON p.product_id=op.product_id
                      left outer join Categories ct on p.category=ct.category_id
                      left outer join Clients c ON c.client_id=o.client_id
WHERE o.order_id=1
/* Zapytanie pokazujące numer oraz datę zamówienia, imię oraz nazwisko klienta oraz
nazwisko przypisanego do zamówienia pracownika, kategorię oraz nazwę i cenę produktu,
a także ilość zamówionych poszczególnych produktów i ich sumaryczną cenę */

SELECT o.order_id, sum(op.quantity*p.price) as total_value
FROM Orders o left outer join Ord_Prod op on o.order_id=op.order_id
                      left outer join Products p on p.product_id=op.product_id
                      WHERE o.order_id=1
                      GROUP BY o.order_id
/* Zapytanie podsumowujące całkowity koszt zamówienia*/
```