JEGYZŐKÖNYV

Adatbázis rendszerek MSc

2021. tavasz féléves feladat

Készítette: **Megyeri Balázs**
Neptunkód: **AXQB0Z**

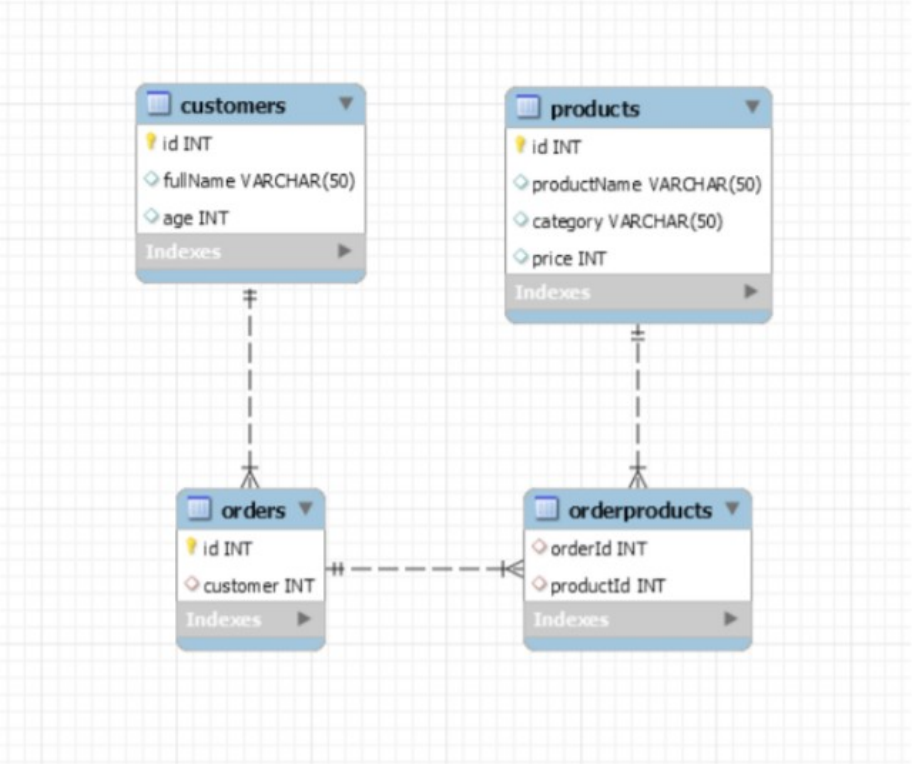
Adatbázis rendszerek Msc

Gyakorlat

2021.03.30

1. feladat

Adott az alábbi ER-modell:



Készítsünk MySQL adatbázist hozzá: hozzuk létre a táblákat és töltsük fel őket adatokkal.

Végezzük el az adatbázison a következő lekérdezéseket LINQ segítségével C# nyelven:

1. Írassuk ki az összes vevő nevét és életkorát életkor szerinti növekvő sorrendben!
2. Írassuk ki a 28 évnél idősebb vevőket név szerint csökkenő sorrendben!
3. Írassuk ki azokat a termékeket, melyek kategóriája tartalmazza a "ház" szót!
4. Keressük meg a legdrágább termékeket és írassuk ki a termékek nevét és árát!
5. Mely termékek ára alacsonyabb a termékek átlag áránál?
6. Keressük meg hány "divat" kategóriájú termék van!
7. Írassuk ki minden rendeléshez a hozzá tartozó termékek árának összegét!
8. Írassuk ki, hogy ki hány darab terméket rendelt!

Megoldás

1.

```
var q1 = Customers.Select(c => new {name = c.FullName, age = c.Age}).OrderBy(c => c.age);
q1.Dump();
```

Query<> (6 items)

name	age
Kiss István	23
Teszt Eszter	26
Balogh Vivien	30
Nagy Károly	45
Varga Zoltán	50
Test John	53
	227

2.

```
var q2 = Customers.Where(c => c.Age > 28).OrderByDescending(c => c.FullName);
q2.Dump();
```

Query<Customers> (4 items)

Id	FullName	Age	CustomerOrders
4	Varga Zoltán	50	CustomerOrders
54	Test John	53	CustomerOrders
2	Nagy Károly	45	CustomerOrders
5	Balogh Vivien	30	CustomerOrders
		178	

3.

```
var q3 = Products.Where(p => p.Category.Contains("ház"));
q3.Dump();
```

Query<Products> (3 items)

Id	ProductName	Category	Price
1	mosógép	háztartási eszköz	30000
3	kenyérsütő	háztartási eszköz	20000
4	hűtő	háztartási eszköz	150000
			200000

4.

```
var maxValue = Products.Max(p => p.Price);
var q4 = Products
            .Where(p => p.Price == maxValue)
            .Select( p => new {
                productName = p.ProductName, price = p.Price
            });
q4.Dump();
```

Query<> (1 item)

productName	price
hűtő	150000

5.

```
//5. feladat Mely termékek ára alacsonyabb a termékek átlag áránál?
var avgPrice = Math.Round((double) Products.Average(p => p.Price));

var q5 = Products
            .Where(p => p.Price < avgPrice)
            .Select( p => new {
                productName = p.ProductName, price = p.Price
            });
q5.Dump();
```

Query<> (6 items)

productName	price
mosógép	30000
fülhallgató	4000
kenyérsütő	20000
sapka	1500
sál	2300
Test name	5000
	62800

6.

```
var q6 = Products.Where(p => p.Category == "divat").Count();
q6.Dump();
```

```
3
```

7.

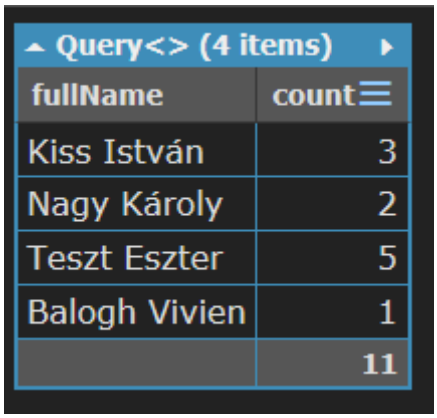
```
var q7 = Orders
            .Join(Orderproducts, o => o.Id, op => op.OrderId, (o, op) => new {o, op})
            .Join(Products, oop => oop.op.ProductId, p => p.Id, (oop, p) => new {oop, p})
            .GroupBy(result => result.oop.o.Id)
            .Select(
                groups => new {
                    orderId = groups.Key,
                    count = groups.Select(g => g.p.Price).Aggregate((p1, p2) => (p1 + p2))
                }
            );
q7.Dump();
```

Query<> (5 items)

orderId	count
1	24000
2	30000
3	7800
4	150000
5	55500
	267300

8.

```
var q8 = Customers
            .Join(Orders, c => c.Id, o => o.Customer, (c, o) => new {c, o})
            .Join(Orderproducts, oc => oc.o.Id, op => op.OrderId, (oc, op) => new {oc, op})
            .Join(Products, ocop => ocop.op.ProductId, p => p.Id, (ocop, p) => new {ocop, p})
            .GroupBy(result => result.ocop.oc.c.FullName)
            .Select(groups => new {fullName = groups.Key, count = groups.Count()});
q8.Dump();
```



Query<> (4 items)

fullName	count
Kiss István	3
Nagy Károly	2
Teszt Eszter	5
Balogh Vivien	1
	11

2. feladat

Készítsünk egyszerű C# konzol alkalmazást az **1. feladat** adatbázisához!

Követelmények:

1. Vevők és termékek adatbázisban való elmentése és törlése.
2. Rendelések felvétele.
3. Lekérdezések:
   a) Összes vevő
   b) Összes termék
   c) Összes rendelés
   d) Felhasználótól bekért kereső kifejezés alapján termékek listázása.
   e) Felhasználótól bekért azonosítójú vevő összes rendelt termékének lekérdezése.

Megoldás

Menü

```
void Main()
{
    while(true) {
        Console.WriteLine();
        Console.WriteLine("--Menu---");
        Console.WriteLine("Add new customer (1)");
        Console.WriteLine("Add new product (2)");
        Console.WriteLine("Create new order (3)");
        Console.WriteLine("Delete customer (4)");
        Console.WriteLine("Delete product (5)");
        Console.WriteLine("List customers (6)");
        Console.WriteLine("List products (7)");
        Console.WriteLine("List orders (8)");
        Console.WriteLine("Search in products (9)");
        Console.WriteLine("Find products by customer (10)");
        Console.WriteLine("Quit (11)");
        Console.WriteLine();
```

```
    string menuInput = Util.ReadLine("What would you like to do?");

    switch(menuInput) {
        case "1" :
            addNewCustomer();
            break;
        case "2":
            addNewProduct();
            break;
        case "3":
            addNewOrder();
            break;
        case "4":
            deleteCustomer();
            break;
        case "5":
            deleteProduct();
            break;
        case "6":
            listCustomers();
            break;
        case "7":
            listProducts();
            break;
        case "8":
            listOrders();
            break;
        case "9":
            searchInProducts();
            break;
        case "10":
            findProductsByCustomer();
            break;
        case "11":
            Console.WriteLine("Bye.");
            return;
        default:
            Console.WriteLine("Invalid command.");
            break;
    }
}
```

1.

```
void addNewCustomer() {
    try {
        Console.WriteLine("Add new customer.");

        int id = int.Parse(Util.ReadLine("Id:"));
        string fullName = Util.ReadLine("FullName:");
        int age = int.Parse(Util.ReadLine("Age:"));

        Customers customer = new Customers { Id = id, FullName = fullName, Age = age };

        Customers.InsertOnSubmit(customer);
        SubmitChanges();

        Console.WriteLine("Customer successfully created.");
    } catch(Exception e) {
        Console.WriteLine("Failed to create customer: " + e.Message);
    }
}
```

2.

```
void addNewOrder() {
    try {
        Console.WriteLine("Create new order.");

        int orderId = int.Parse(Util.ReadLine("OrderId:"));
        int customerId = int.Parse(Util.ReadLine("CustomerId:"));
        string productIdsString = Util.ReadLine("ProductIds(separated by comma):");
        string[] separatedProductIdsString = productIdsString.Split(',');
        int[] productIds = Array.ConvertAll(separatedProductIdsString, s => int.Parse(s));

        Orders order = new Orders { Id = orderId, Customer = customerId };
        Orders.InsertOnSubmit(order);

        foreach (int productId in productIds) {
            Orderproducts orderProduct = new Orderproducts {OrderId = orderId, ProductId = productId};
            Orderproducts.InsertOnSubmit(orderProduct);
        }

        SubmitChanges();

        Console.WriteLine("Order successfully created.");
    } catch(Exception e) {
        Console.WriteLine("Failed to create order: " + e.Message);
    }
}
```

```
void deleteCustomer() {
    int customerId = int.Parse(Util.ReadLine("CustomerId:"));

    var customer = Customers.Where(c => c.Id == customerId).First();

    Customers.DeleteOnSubmit(customer);

    SubmitChanges();

    Console.WriteLine("Customer successfully removed.");
}

void deleteProduct() {
    int productId = int.Parse(Util.ReadLine("ProductId:"));

    var product = Products.Where(p => p.Id == productId).First();

    Products.DeleteOnSubmit(product);

    SubmitChanges();

    Console.WriteLine("Product successfully removed.");
}

void listCustomers() {
    var queryResults = Customers.Select(c => new {fullName = c.FullName, age = c.Age});

    Console.WriteLine("Customers:");

    queryResults.Dump();
}
```

3.

```
void listProducts() {
    var queryResults = Products.Select(p => new {
        productName = p.ProductName,
        category = p.Category,
        price = p.Price
    });

    Console.WriteLine("Products:");

    queryResults.Dump();
}
```

```
void searchInProducts() {
    string searchString = Util.ReadLine("Search in products:");

    var queryResults = Products.Where(
        p => p.Category.Contains(searchString) || p.ProductName.Contains(searchString)
    );

    queryResults.Dump();
}

void findProductsByCustomer() {
    int customerId = int.Parse(Util.ReadLine("CustomerId:"));

    var queryResults = Customers
            .Where(c => c.Id == customerId)
            .Join(Orders, c => c.Id, o => o.Customer, (c, o) => new {c, o})
            .Join(Orderproducts, oc => oc.o.Id, op => op.OrderId, (oc, op) => new {oc, op})
            .Join(Products, ocop => ocop.op.ProductId, p => p.Id, (ocop, p) => new {ocop, p})
            .GroupBy(result => result.ocop.oc.c.FullName)
            .Select(groups => new {
                fullName = groups.Key,
                products = groups.Select(g => new {
                    orderId = g.ocop.oc.o.Id,
                    productId = g.p.Id,
                    productName = g.p.ProductName
                })
            });

    queryResults.Dump();
}
        }

        Console.WriteLine();
    }
}
```