Kapcsolat-orientált adatelérés megvalósítása C#-ban, Access adatbázissal

0. `using System.Data.OleDb;`

1. Kapcsolat beállítása:
   a. .mdb fájlok esetén:

```
static string conn=@"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=adatbazis.mdb";
static public OleDbConnection kapcsolat = new OleDbConnection(conn);
```

   b. .accdb fájlok esetén:

```
static public string conn=@"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=adatbazis.accdb";
static public OleDbConnection kapcsolat = new OleDbConnection(conn);
```

2. Parancsobjektum létrehozása:

```
OleDbCommand parancs = new OleDbCommand();
parancs.Connection = kapcsolat;
parancs.CommandText = "SELECT * FROM lemezbolt";
```

3. Kapcsolat megnyitása:

```
kapcsolat.Open();
```

4. Parancs végrehajtása

   a. SELECT utasítás végrehajtása, amelynek az eredménye rekordok halmaza:

```
OleDbDataReader dreader = parancs.ExecuteReader();
```

   Egy sor kiolvasása:

```
dreader.Read();
```

   A beolvasott sor oszlopainak kinyerése:

```
dreader["mezőnév"] vagy reader[index] alakban
```

   Ha minden sor ki akarunk nyerni:

```
while (dreader.Read())
{
}
```

   Mivel nincs olyan tulajdonsága a datareader objektumnak, amiből a kinyert sorok számát le tudnánk kérdezni, ezért ha erre is szükségünk van, a while ciklusban meg kell számolni:

```
int n = 0;
while (dreader.Read())
{
  n++;
}
```

   b. INSERT, UPDATE, DELETE utasítás végrehajtása:

```
parancs.ExecuteNonQuery();
```

   Az előbbi függvény visszaadja az érintett rekordok számát is, ezért ha erre is szükségünk van:

```
int n= (int) parancs.ExecuteNonQuery();
```

   c. Olyan SELECT utasítás esetén, melynek egyetlen (skalár) visszaadott értéke van (pl. AVG()):

```
int n= parancs.ExecuteScalar();
```

5. DataReader bezárása (ha volt), kapcsolat bezárása – minden esetben:

```
dreader.Close();
kapcsolat.Close();
```