C# en MySQL

Inleiding

In deze les wordt geleerd hoe je een **MySQL** database kunt gebruiken met **C#**.
Hiervoor heb je nodige een programma die lokaal een MySQL server aanmaakt op de computer. Bijvoorbeeld **MAMP** of **WAMP**. Ook zou je direct een koppeling kunnen maken met een actieve MySQL database.
vorige lessen zijn de termen class en object al enkele keren genoemd, maar er is nog niet concreet over deze termen gesproken. Dat gaat in d les gebeuren.

Installatie procedure

Stap 1:

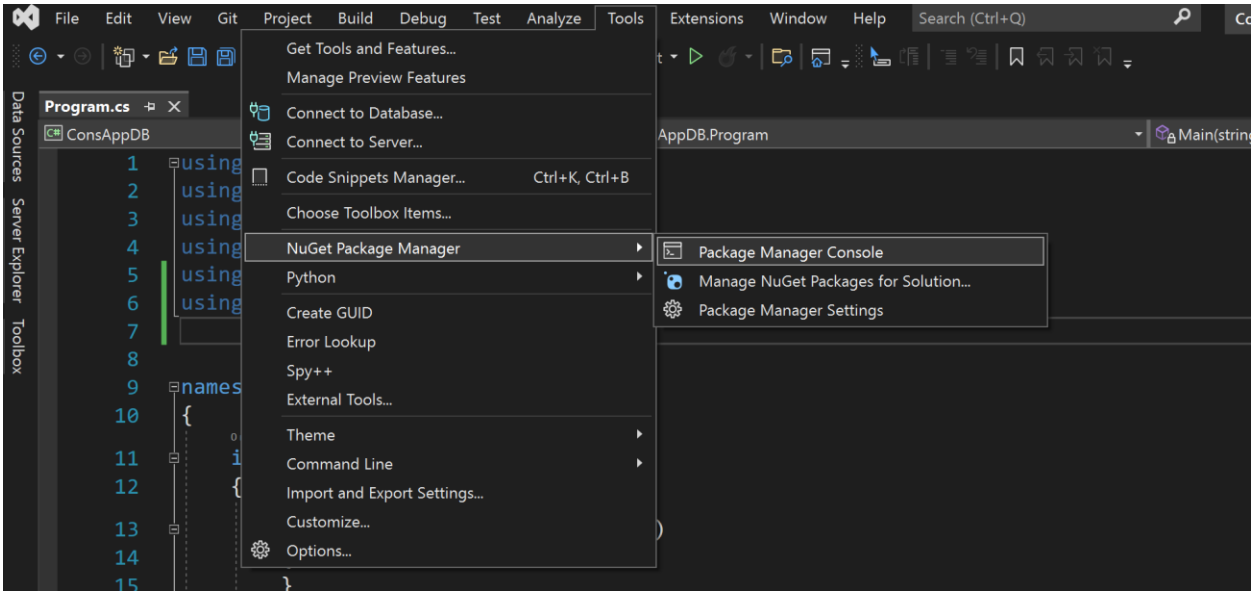
Start **Visual Studio Community (VSC)** en maak een **Console Applicatie** met **.NET framework** aan.

Stap 2:

Wanneer VSC gestart is kies de volgende menu opties:

1. Tools
2. NuGet Package Manager
3. Package Manager Console

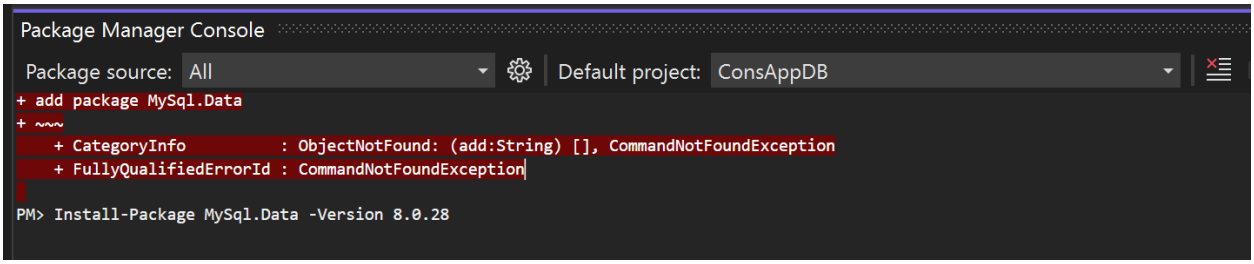
Zie onderstaande afbeelding:



C# en MySQL

Stap 3:

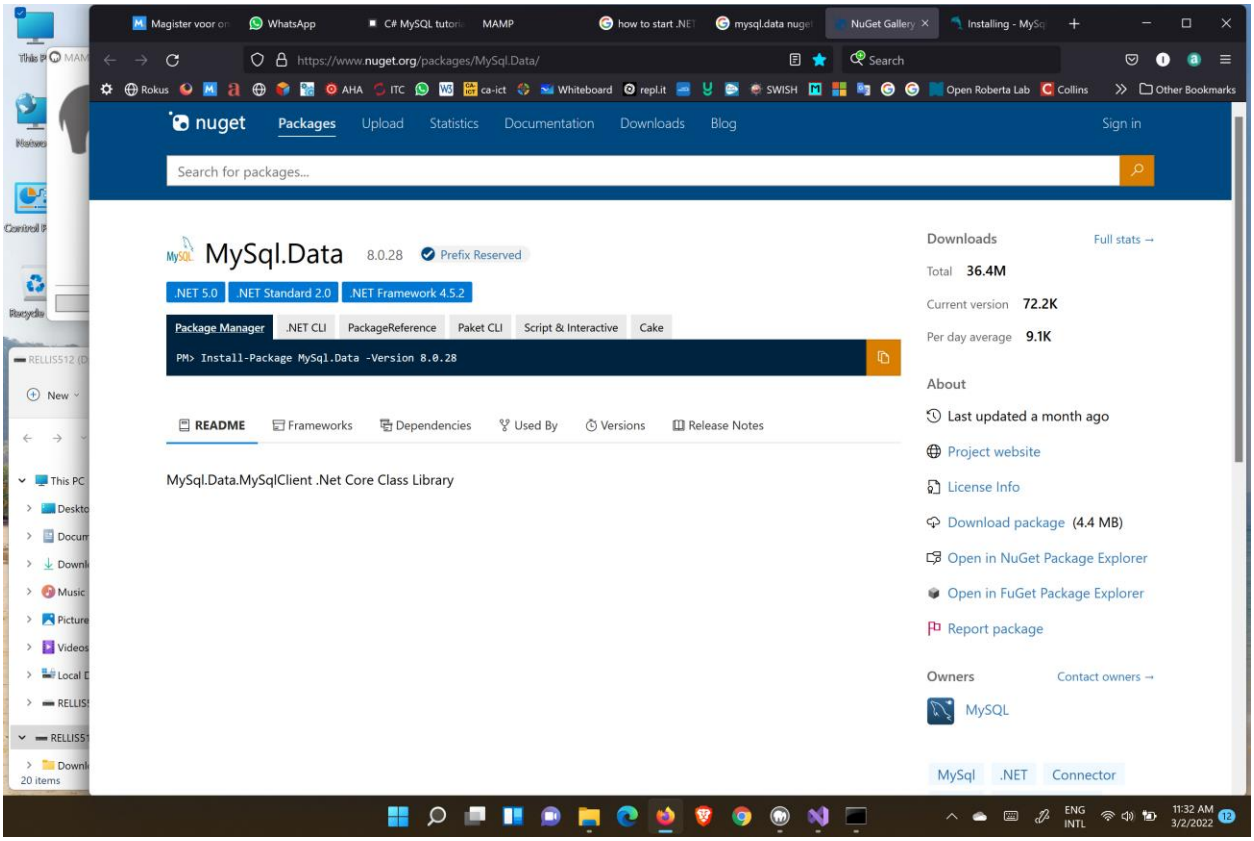
In de Package Manager Console *(zie onderstaande afbeelding)*,



voer de volgende tekst bij de prompt in:

PM> **Install-Package MySql.Data -Version 8.0.28**

Ga eerst naar de volgende link, https://www.nuget.org/packages/MySql.Data/, om te kijken welke versie de laatste is.



Opmerking:
Deze stappen moeten voor elk C3 Project opnieuw worden herhaald.

C# en MySQL

MySQL

MySql.Data is een implementatie van de **ADO.NET**-specificatie voor de MySQL-database. Het is een stuurprogramma geschreven in C#-taal en is beschikbaar voor alle .NET-talen.

```
$ dotnet add package MySql.Data
```

We nemen het pakket op in ons .NET Core-project.

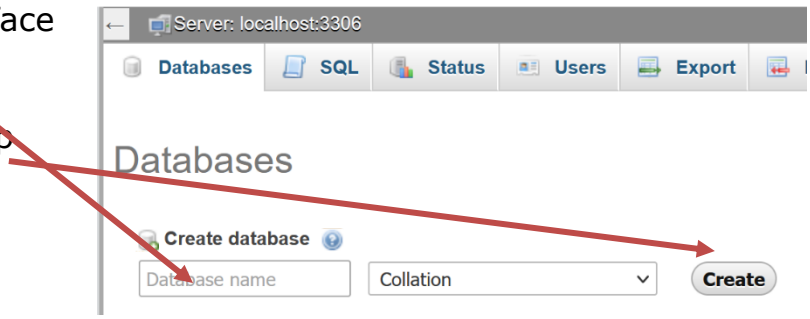
De **`MySqlConnection`**, **`MySqlCommand`**, **`MySqlDataReader`**, **`DataSet`** en **`MySqlDataProvider`** zijn de kernelementen van het .NET-gegevensprovidermodel. De `MySqlConnection` maakt een verbinding met een specifieke gegevensbron. Het `MySqlCommand`-object voert een SQL-instructie uit op een gegevensbron. De `MySqlDataReader` leest gegevensstromen uit een gegevensbron.

Het `DataSet`-object wordt gebruikt voor offline werk met een grote hoeveelheid gegevens. Het is een niet-verbonden gegevensrepresentatie die gegevens uit verschillende bronnen kan bevatten. Zowel `MySqlDataReader` als `DataSet` worden gebruikt om met gegevens te werken; ze worden onder verschillende omstandigheden gebruikt. Als we alleen de resultaten van een query hoeven te lezen, is de `MySqlDataReader` de betere keuze. Als we meer uitgebreide gegevensverwerking nodig hebben, of als we een **Winforms**-besturingselement aan een databasetabel willen binden, heeft de `DataSet` de voorkeur.

Voorbeeld code

Het onderstaande programma bepaald de MySQL vesie in C#. Voor dat je dit programma uitvoert moet eerst:

1. een MySQL server zijn opgestart. Dit kan zijn doormiddel van MAMP, WAMP of ander verwant MySQL server programma.
2. een test database zijn aangemaakt met bijvoorbeeld de **phpMyAdmin** interface van MySQL.
    a. Vul naam in
    b. Klik op de knop Create



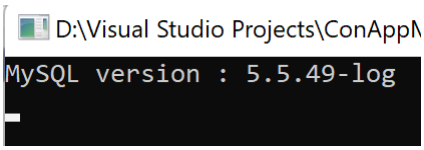
C# en MySQL

```
Program.cs
ConAppMySQL_1 | ConAppMySQL_1.Program | Main(string[] args)

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient;
7
8  namespace ConAppMySQL_1
9  {
       0 references
10     internal class Program
11     {
           0 references
12         static void Main(string[] args)
13         {
14             string cs = @"server=localhost;userid='root';password='root';database='testdb'";
15
16             var con = new MySqlConnection(cs);
17             con.Open();
18
19             Console.WriteLine($"MySQL version : {con.ServerVersion}");
20
21             Console.ReadKey();
22         }
23     }
24 }
25
```

In dit voorbeeld is gebruik gemaakt van MAMP v3.2.2 voor Windows. Dit is een ouder versie van MAMP.

De standaard gebruikersnaam *(default username)* is 'root' en het standaard wachtwoord is 'root'

Wanneer men dit programma uitvoert wordt in een Console venster de versie van de geïnstalleerde MySQL server getoond.
Het programma maakt verbinding met de database en krijgt wat informatie over de MySQL-server.



Uitleg van de code:

De onderstaande instructie importeert de elementen van de MySQL-gegevensprovider.

```
using MySql.Data.MySqlClient;
```

C# en MySQL

De onderstaande instructie is de verbindingsreeks (***Connection string***). Het wordt door de gegevensprovider gebruikt om een verbinding met de database tot stand te brengen. We specificeren de hostnaam, gebruikersnaam, wachtwoord en een databasenaam.

```
string cs = @"server=localhost;userid=dbuser;password=s$cret;database=testdb";
```

Er wordt een **MySQLConnection**-object gemaakt doormiddel van de onderstaande instructie. Dit object wordt gebruikt om een verbinding met een database te openen.

```
var con = new MySqlConnection(cs);
```

De volgende regel opent de databaseverbinding.

```
con.Open();
```

Met de volgende instructie drukken we de versie van MySQL af met behulp van de eigenschap ServerVersion van het verbindingsobject.

```
Console.WriteLine($"MySQL version : {con.ServerVersion}");
```

De MySQL SELECT-instructie in C#

Het onderstaande programma laat de MySQL versie zien door gebruik te maken van de SELECT-instructie van MySQL.

Program.cs | ConAppMySQL-2 | ConAppMySQL_2.Program | Main(string[] args)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace ConAppMySQL_2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string cs = @"server=localhost;userid='root';password='root';database='testdb'";

            var con = new MySqlConnection(cs);
            con.Open();

            var stm = "SELECT VERSION()";
            var cmd = new MySqlCommand(stm, con);

            var version = cmd.ExecuteScalar().ToString();
            Console.WriteLine($"MySQL version: {version}");

            Console.ReadKey();
        }
    }
}
```

Uitleg van de code:

We controleren op de versie van de MySQL-database. Dit keer met behulp van een SQL-query.

```
var stm = "SELECT VERSION()";
```

Dit is de SQL SELECT-instructie. Het geeft de versie van de database terug. De VERSIE is een ingebouwde MySQL-functie.

De `MySqlCommand` in de onderstaande instructie is een object dat wordt gebruikt om een query op de database uit te voeren. De parameters zijn de SQL-instructie en het verbindingsobject.

```
var cmd = new MySqlCommand(stm, con);
```

Er zijn query's die alleen een scalaire waarde (*maat in getal vorm*) retourneren. In ons geval willen we een eenvoudige string die de versie van de database specificeert. De ExecuteScalar wordt in dergelijke situaties gebruikt.

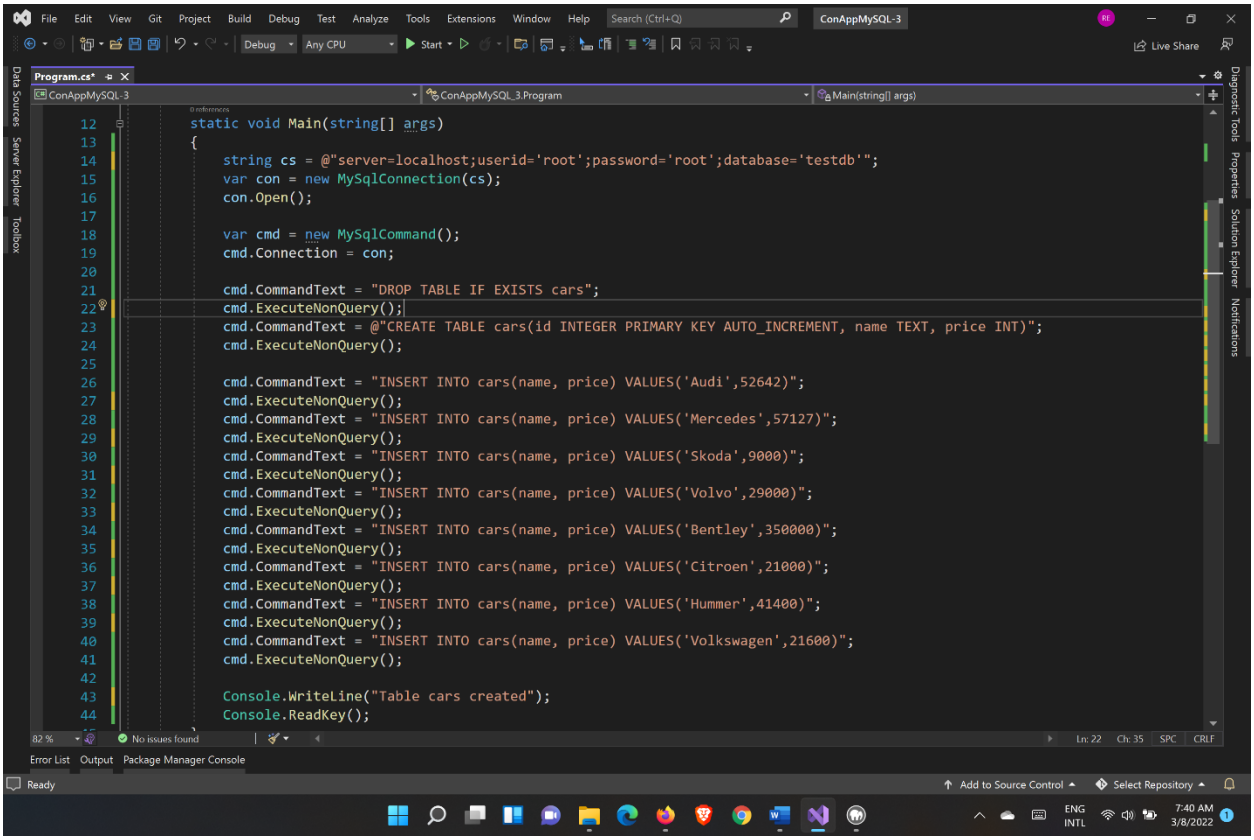
```
var version = cmd.ExecuteScalar().ToString();
```

C# en MySQL

De MySQL CREATE TABLE-instructie in C#

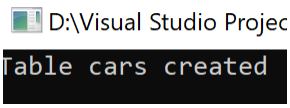
De onderstaande code laat zien hoe je een tabel in de database kunt aanmaken en records aan e tabel kunt toevoegen.

Opmerking: Vergeet niet eerst je MySQL server applicatie (MAMP, WAMP) op te starten voor je het programma uitvoert.



```
static void Main(string[] args)
{
    string cs = @"server=localhost;userid='root';password='root';database='testdb'";
    var con = new MySqlConnection(cs);
    con.Open();

    var cmd = new MySqlCommand();
    cmd.Connection = con;

    cmd.CommandText = "DROP TABLE IF EXISTS cars";
    cmd.ExecuteNonQuery();
    cmd.CommandText = @"CREATE TABLE cars(id INTEGER PRIMARY KEY AUTO_INCREMENT, name TEXT, price INT)";
    cmd.ExecuteNonQuery();

    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Audi',52642)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Mercedes',57127)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Skoda',9000)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Volvo',29000)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Bentley',350000)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Citroen',21000)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Hummer',41400)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Volkswagen',21600)";
    cmd.ExecuteNonQuery();

    Console.WriteLine("Table cars created");
    Console.ReadKey();
}
```

Dit programma maakt de cars tabel aan met 8 rijen aan data. Na de uitvoer volgt de onderstaande meldig als je alle goed gedaan hebt.



Uitleg van de code:

Eerst wordt er gecontroleerd of de tabel al bestaat. Indien dit het geval is wordt deze verwijderd met de methode **ExecuteNonQuery** als we geen resultatenset willen, bijvoorbeeld voor **DROP-**, **INSERT**- of **DELETE**-instructies.

```
cmd.CommandText = "DROP TABLE IF EXISTS cars";
cmd.ExecuteNonQuery();
```

Met de volgende instructie wordt de tabel cars aan gemaakt. Het sleutelwoord **AUTOINCREMENT** zorgt ervoor dat de kolom automatisch wordt verhoogd in MySQL.

```
cmd.CommandText = @"CREATE TABLE cars(id INTEGER PRIMARY KEY AUTO_INCREMENT,
        name TEXT, price INT)";
cmd.ExecuteNonQuery();
```

Vervolgens worden er records (rijen) aan de tabel toegevoegd. Hieronder zie je de eerste twee instructies. De andere zijn identiek alleen met andere waarden in de data velden.

```
cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Audi',52642)";
cmd.ExecuteNonQuery();

cmd.CommandText = "INSERT INTO cars(name, price) VALUES('Mercedes',57127)";
cmd.ExecuteNonQuery();
...
```

In de phpMyAdmin is te controleren of de data inderdaad is toegevoegd aan de tabel.

		id	name	price
☐	Edit Copy Delete	1	Audi	52642
☐	Edit Copy Delete	2	Mercedes	57127
☐	Edit Copy Delete	3	Skoda	9000
☐	Edit Copy Delete	4	Volvo	29000
☐	Edit Copy Delete	5	Bentley	350000
☐	Edit Copy Delete	6	Citroen	21000
☐	Edit Copy Delete	7	Hummer	41400
☐	Edit Copy Delete	8	Volkswagen	21600

C# en MySQL

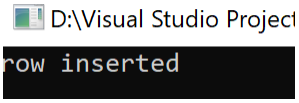
De MySQL Voorbereide-instructie in C#

Voorbereide instructies verhogen de veiligheid en prestaties. Wanneer we voorbereide instructies schrijven, gebruiken we tijdelijke aanduidingen in plaats van de waarden rechtstreeks in de instructies te schrijven.

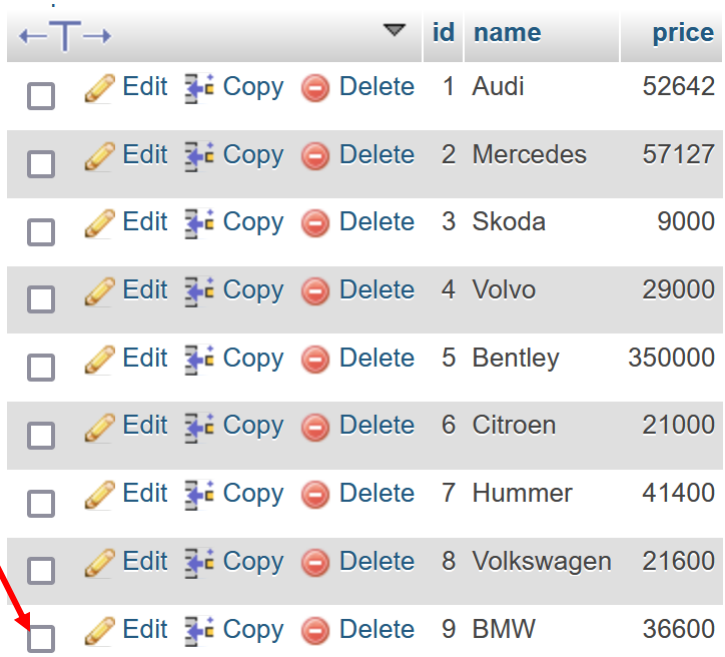
Bekijk de onderstaande code fragment:

```
12      static void Main(string[] args)
13      {
14          string cs = @"server=localhost;userid='root';password='root';database='testdb'";
15
16          var con = new MySqlConnection(cs);
17          con.Open();
18
19          var sql = "INSERT INTO cars(name, price) VALUES(@name, @price)";
20          var cmd = new MySqlCommand(sql, con);
21
22          cmd.Parameters.AddWithValue("@name", "BMW");
23          cmd.Parameters.AddWithValue("@price", 36600);
24          cmd.Prepare();
25
26          cmd.ExecuteNonQuery();
27
28          Console.WriteLine("row inserted");
29
30          Console.ReadKey();
31      }
```

Na uitvoer wordt in het Console venster getoond dat de rij is toegevoeg.



In phpMyAdmin zie dat er een 9de rij is toegevoegd nadat je de pagina ververst.



	id	name	price
Edit Copy Delete	1	Audi	52642
Edit Copy Delete	2	Mercedes	57127
Edit Copy Delete	3	Skoda	9000
Edit Copy Delete	4	Volvo	29000
Edit Copy Delete	5	Bentley	350000
Edit Copy Delete	6	Citroen	21000
Edit Copy Delete	7	Hummer	41400
Edit Copy Delete	8	Volkswagen	21600
Edit Copy Delete	9	BMW	36600

Uitleg van de code:

We voegen een nieuwe auto toe aan de tabel met auto's. We gebruiken een geparametriseerd commando.

```
var sql = "INSERT INTO cars(name, price) VALUES(@name, @price)";
using var cmd = new MySqlCommand(sql, con);
```

Wanneer we voorbereide instructies schrijven, gebruiken we tijdelijke aanduidingen in plaats van de waarden rechtstreeks in de instructies te schrijven. Voorbereide instructies zijn sneller en beschermen tegen SQL-injectieaanvallen. De @name en @price zijn tijdelijke aanduidingen, die later worden ingevuld.

```
cmd.Parameters.AddWithValue("@name", "BMW");
cmd.Parameters.AddWithValue("@price", 36600);
cmd.Prepare();
```

Waarden zijn gebonden aan de tijdelijke aanduidingen met de `AddWithValue`-methode.

De voorbereide instructie wordt uitgevoerd. We gebruiken de `ExecuteNonQuery`-methode van het MySQLCommand-object wanneer we niet verwachten dat er gegevens worden geretourneerd.

```
cmd.ExecuteNonQuery();
```

C# en MySQL

De MySQLDataReader-instructie in C#

De **MySqlDataReader** is een object dat wordt gebruikt om gegevens uit de database op te halen. Het biedt snelle, alleen-vooruit-lezen toegang tot queryresultaten. Het is de meest efficiënte manier om gegevens uit tabellen op te halen.

In het onderstaande programma worden alle rijen uit de tabel cars opgehaald en getoond in het Console venster.

```
namespace ConAppMySQL_5
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string cs = @"server=localhost;userid='root';password='root';database='testdb'";

            var con = new MySqlConnection(cs);
            con.Open();

            string sql = "SELECT * FROM cars";
            var cmd = new MySqlCommand(sql, con);

            MySqlDataReader rdr = cmd.ExecuteReader();

            while (rdr.Read())
            {
                Console.WriteLine("{0} {1} {2}", rdr.GetInt32(0), rdr.GetString(1),
                        rdr.GetInt32(2));
            }
        }
    }
}
```

De uitvoer ziet er als volgt uit in het Console venster.

```
C:\windows\system32\cmd.exe
1 Audi 52642
2 Mercedes 57127
3 Skoda 9000
4 Volvo 29000
5 Bentley 350000
6 Citroen 21000
7 Hummer 41400
8 Volkswagen 21600
9 BMW 36600
Press any key to continue . . .
```

Uitleg van de code:

Om een **SQLDataReader** te maken, noemen we de `ExecuteReader`-methode van het SqlCommand-object.

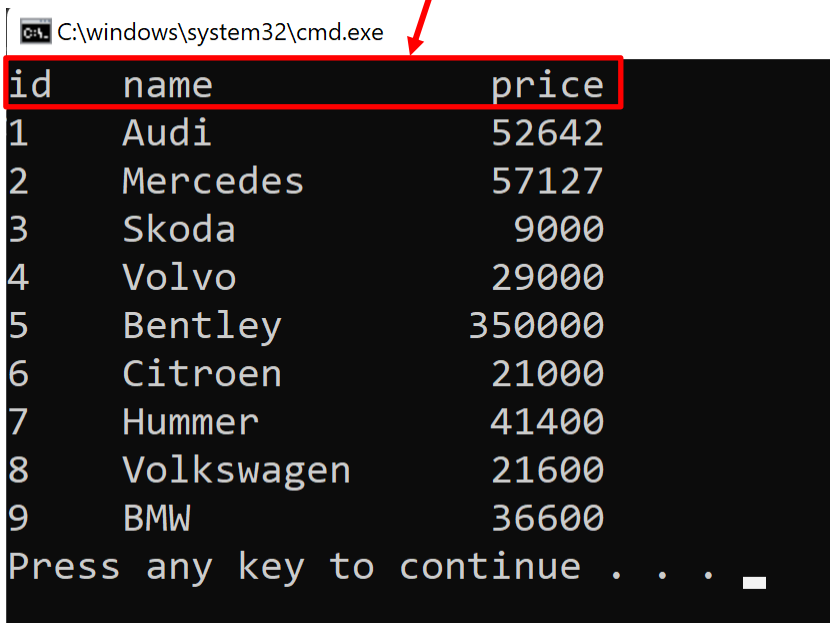
```
using MySqlDataReader rdr = cmd.ExecuteReader();
```

Met de `Read`-methode gaat de gegevenslezer naar het volgende record. Het geeft *true* terug als er meer rijen zijn; anders *false*. We kunnen de waarde ophalen met behulp van de matrix-index-notatie, of een specifieke methode gebruiken om toegang te krijgen tot kolomwaarden in hun oorspronkelijke gegevenstypen. Dat laatste is efficiënter.

```
while (rdr.Read())
{
    Console.WriteLine("{0} {1} {2}", rdr.GetInt32(0), rdr.GetString(1),
            rdr.GetInt32(2));
}
```

MySQL kolomkoppen (*headers*) in C#

In het volgende voorbeeld drukken we **kolomkoppen** (*headers*) af met de gegevens uit een databasetabel.



Hieronder staat een afbeelding van de code:

```
12      static void Main(string[] args)
13      {
14          string cs = @"server=localhost;userid='root';password='root';database='testdb'";
15
16          var con = new MySqlConnection(cs);
17          con.Open();
18
19          var sql = "SELECT * FROM cars";
20
21          var cmd = new MySqlCommand(sql, con);
22
23          MySqlDataReader rdr = cmd.ExecuteReader();
24          Console.WriteLine($"{rdr.GetName(0),-4} {rdr.GetName(1),-10} {rdr.GetName(2),10}");
25
26          while (rdr.Read())
27          {
28              Console.WriteLine($"{rdr.GetInt32(0),-4} {rdr.GetString(1),-10} {rdr.GetInt32(2),10}");
29          }
30      }
```

Uitleg van de code:

In het voorbeeld selecteren we alle rijen uit de tabel cars met hun kolomnamen.

```
var sql = "SELECT * FROM cars";
var cmd = new MySqlCommand(sql, con);
```

We krijgen de namen van de kolommen met de `GetName`-methode van de lezer.

```
Console.WriteLine($"{rdr.GetName(0),-4} {rdr.GetName(1),-10} {rdr.GetName(2),10}");
```

We printen de gegevens die door de SQL-instructie zijn geretourneerd naar het Console venster met de onderstaande code:

```
while (rdr.Read())
{
    Console.WriteLine($"{rdr.GetInt32(0),-4} {rdr.GetString(1),-10}
{rdr.GetInt32(2),10}");
}
```

```
C:\windows\system32\cmd.exe
id   name            price
1    Audi            52642
2    Mercedes        57127
3    Skoda            9000
4    Volvo           29000
5    Bentley        350000
6    Citroen         21000
7    Hummer          41400
8    Volkswagen      21600
9    BMW             36600
Press any key to continue . . .
```