Databases

Sql vs NoSql

- SQL - Structured Query Language
- RDBMS - Relational_database_management_system (MSSql, PostgreSql, MySql, Oracle), etc
- NOSQL
    - column: cassandra, hbase,
    - document: mongoDB,
    - key-value: redis,
    - graph: Neo4j

Пара интересных докладов с highload: postgresql worst practices, cassandra успехи и провалы

Ado.net

- Интерфейс взаимодействия с rdbms
  - Connection
  - Command
  - DataReader
  - DataSet
  - DataAdapter

```
string connectionString = "Data Source=(local);Initial
Catalog=Northwind;Integrated Security=true";
string queryString =
  "SELECT ProductID, UnitPrice, ProductName"
  + "FROM dbo.products "
  + "WHERE UnitPrice > @pricePoint "
  + "ORDER BY UnitPrice DESC;";
int paramValue = 5;
using (SqlConnection connection = new SqlConnection(connectionString))
{
  SqlCommand command = new SqlCommand(queryString, connection);
  command.Parameters.AddWithValue("@pricePoint", paramValue);

  connection.Open();
  SqlDataReader reader = command.ExecuteReader();
  while (reader.Read())
  {
      Console.WriteLine("\t{0}\t{1}\t{2}", reader[0], reader[1], reader[2]);
  }
  reader.Close();
}
```

- ExecuteNonQuery: просто выполняет sql-выражение и возвращает количество измененных записей. Подходит для sql-выражений INSERT, UPDATE, DELETE.
- ExecuteReader: выполняет sql-выражение и возвращает строки из таблицы. Подходит для sql-выражения SELECT.
- ExecuteScalar: выполняет sql-выражение и возвращает одно скалярное значение, например, число. Подходит для sql-выражения SELECT в паре с одной из встроенных функций SQL, как например, Min, Max, Sum, Count.

```
string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=usersdb;Integrated Security=True";
string sqlExpression = "INSERT INTO Users (Name, Age) VALUES ('Tom', 18)";

using (SqlConnection connection = new SqlConnection(connectionString))
{
  connection.Open();
  SqlCommand command = new SqlCommand(sqlExpression, connection);
  int number = command.ExecuteNonQuery();
}
```

```
string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=usersdb;Integrated Security=True";
string sqlExpression = "SELECT * FROM Users";

using (SqlConnection connection = new SqlConnection(connectionString))
{
  connection.Open();
  SqlCommand command = new SqlCommand(sqlExpression, connection);
  SqlDataReader reader = command.ExecuteReader();

  if(reader.HasRows) // если есть данные
  {
    Console.WriteLine("{0}\t{1}\t{2}", reader.GetName(0), reader.GetName(1),
reader.GetName(2));
    while (reader.Read()) // построчно считываем данные
    {
      int id = reader.GetInt32(0);
      string name = reader.GetString(1);
      int age = reader.GetInt32(2);
    }
  }
```

```
    reader.Close();
}
```

```
string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=usersdb;Integrated Security=True";
string sqlExpression = "SELECT COUNT(*) FROM Users";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    SqlCommand command = new SqlCommand(sqlExpression, connection);
    object count = command.ExecuteScalar();
}
```

```
string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=usersdb;Integrated Security=True";
string sqlExpression = "INSERT INTO Users (Name, Age) VALUES (@name, @age)";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    SqlCommand command = new SqlCommand(sqlExpression, connection);
    SqlParameter nameParam = new SqlParameter("@name", name);
    command.Parameters.Add(nameParam);
    SqlParameter ageParam = new SqlParameter("@age", age);
    command.Parameters.Add(ageParam);
    int number = command.ExecuteNonQuery();
}
```

DBTransaction

```
using (DbConnection connection = database.CreateConnection())
{
  await connection.OpenAsync();
  using (DbTransaction transaction = connection.BeginTransaction(isolationLevel))
  {
    try
    {
      // SqlCommand command = new SqlCommand(sqlExpression, connection,
transaction);
      await process(connection, transaction);
      transaction.Commit();
    }
    catch (Exception processException)
    {
      try
      {
        transaction.Rollback();
      }
      catch (Exception rollbackException)
      {
```

```
            throw new AggregateException(processException, rollbackException);
        }
        throw;
}}}
```

TransactionalScope

```
using(TransactionScope scope = new TransactionScope())
{
  /* Perform transactional work here */
  scope.Complete();
}
```

Dapper

Легковесный маппинг параметров

```
string sql = "SELECT * FROM Invoices";

using (var conn = My.ConnectionFactory())
{
    var invoices = conn.Query<Invoice>(sql);
}
```

Linq2db

Простые мапперы в запросы sql на примере linq2db