Database and Information Systems

Michal Krátký & Pavel Bednář

Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB – Technical University of Ostrava

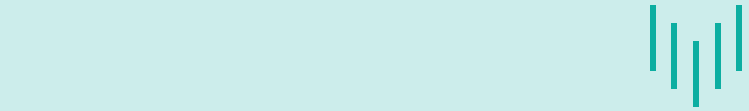
2020/2021

Content

What ORM is?

- **Data layer** – a layer between a database and our program (an information system).
- In general, there are many ways how to implement the data layer of an information system.
- **Object-Relational Mapping (ORM)** is one of them: ORM is a programming technique accessing relation data in a DBMS from an object-oriented programming environment (for example C# or Java).

ORM – tools, frameworks



- There are two techniques to the implementation of ORM:
  - **The 3rd party frameworks:** Hibernate[1], Java Persistence API (JPA)[2], Entity Framework[3].
  - **An own implementation** – or we can implement a generator building a part of ORM.
- In the 3rd party frameworks we have to define a mapping with: a configuration file or an annotation in the source code.
- We then work with a database using an API of a framework.

[1] `http://www.hibernate.org/`
[2] `https://www.oracle.com/technical-resources/articles/java/jpa.html`
[3] `https://docs.microsoft.com/ef/`

Own implementation vs 3rd party frameworks

Advantages:

- the full control of SQL commands,
- we can use features of a specific DBMS (data types, functions, options and so on).

Disadvantages:

- It takes time,
- It is not simply possible to change a DBMS.

Own implementation vs 3rd party frameworks

Advantages:

- The application development is faster,
- In many cases, we can simply change a DBMS.

Disadvantages:

- In many cases, we can not control SQL commands - only a general SQL dialect is used.
- In many cases, we get poor performance. However, there are frameworks specific for a DBMS with a good performance (LINQ[4] pro SQL Server, ADF pro Oracle[5]).

[4] `https://docs.microsoft.com/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries`

[5] `https://www.oracle.com/database/technologies/developer-tools/adf/`

Performance Measurement – Query 1

- Compute the sum in the table $Transfer$ (it includes 1mil. of records), an algorithm:[6]
  1. Send the query: `SELECT * FROM Transfer`
  2. Read the result set and transfer it into a collection of objects stored in the main memory.
  3. Iterate through the objects and compute the sum.

[6] It is only an example of a computation, in a real case we use the SUM aggregation function.

EclipseLink

```
.....
double sum = 0;
List<Transfer> seznam = new ArrayList<Transfer>();
EntityManagerFactory factory =
Persistence.createEntityManagerFactory("Tester2PU");
EntityManager em = factory.createEntityManager();

em = factory.createEntityManager();
Query query = em.createQuery("SELECT p FROM Transfer p");
list = query.getResultList();

for (Transfer element : list)
{
  sum += element.getAmount().doubleValue();
}
.....
```

Hibernate

```
.....
double sum = 0;
Session session = null;
List<Transfer> seznam = new ArrayList<Transfer>();

session = HibernateUtil.getSessionFactory().openSession();
list = (List<Transfer>)session
.createQuery("from␣Transfer").list();

for (Transfer element : list)
{
  sum += element.getAmount().doubleValue();
}
.....
```

Own Implementation

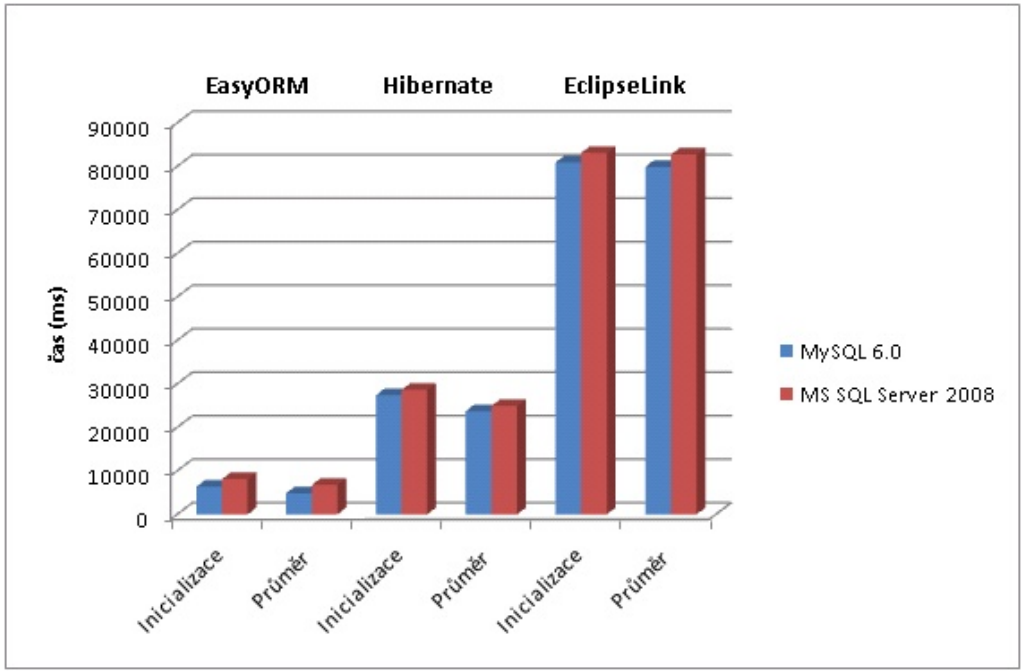
```
.....
double sum = 0;
TransferDB transferDB = new TransferDB();
List<Transfer> list = new ArrayList<Transfer>();

list = transferDB.getAll();

for (Transfer element : list)
{
  sum += element.getAmount();
}
.....
```

Results [7]

	Own ORM		Hibernate		EclipseLink	
	1st Run	Avg.	1st Run	Avg.	1st Run	Avg.
MySQL	6 344	4 868	27 406	23 728	80 922	79 890
MS SQL	8 126	6 752	28 605 **(+250%)**	24 912	83 114 **(+923%)**	82 726



[7] Processing time in miliseconds.

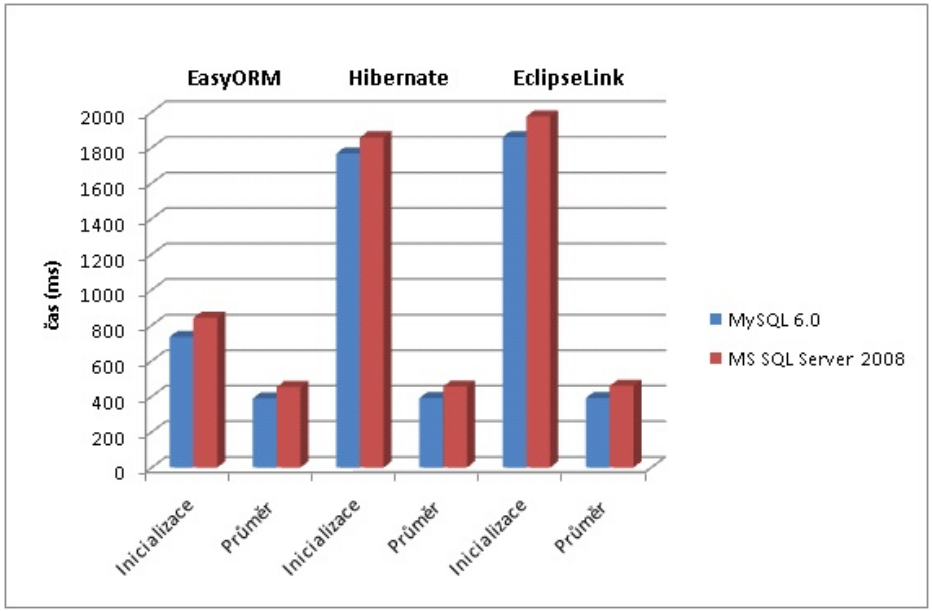
Discussion

- Own ORM is up-to one order of magnitude faster than 3rd party frameworks.
- **Why?** The 3rd party frameworks often include a conceptual layer bewteen a database and an ORM.

Query 2

- The same query as the previous one, however the agggregation SUM function is used: `SELECT SUM(amount) AS sum FROM Transfer`

	Own ORM		Hibernate		EclipseLink	
	1st Run	Avg.	1st Run	Avg.	1st Run	Avg.
MySQL	735	389	1766	390	1859 **(+153%)**	391
MS SQL	842	453	1857 **(+120%)**	456	1976	459



Evaluation

- The 3rd party frameworks provide poor performance especially in the case of queries with low selectivity (i.e. when the query result is large).
- Own ORM is always faster than 3rd party frameworks.
- **The data layer task:** to provide the maximum performance of data access.
- **Result:** when the high throughput of data layer is necessary we have to properly test the performance of an ORM framework selected.

Data Caching

- In the 3rd party frameworks, we can seen that the performance of the first run is much lower compared to other runs, the reason is **data caching** in an internal cache.
- It is often efficient (but the efficiency is similar as the own ORM implementation not using data caching for these data), however it is necessary **to solve updates and transactions out of a DMBS**.

ORM 1/2

- There are many ways how to implement ORM.
- In this subject we are going to use the following mapping:
  1. **One class represents one entity type** (one object represents one record of a table):
     - It is a part of the application layer.
     - Names: **application object**, **domain object**, **Data Transfer Object - DTO**.
     - For example: we create the class `Person` for the table `Person`.
     - In general, we can map one entity type to more domain objects or more entity types on one domain object.

ORM 2/2

- In this course we are going to use the following mapping:
  2. **One class represents one table** (and its operations):
     - A part of the data layer.
     - Name: **Data Access Object - DAO**.
     - In general, one class can include operations for more tables (reports, transactions, ...).
     - For example: we create the class `PersonTable` with static methods Select, Insert, ... for the table `Person`.
  3. **Auxiliary classes** - a management of the connection to a database, the transaction support, ...

Entity

- An entity in ORM is an object which is stored in a database (as one record of a table in many cases).
- Let us have the table `User`:

```
1  CREATE TABLE "User" (
2  idUser          INTEGER NOT NULL PRIMARY KEY,
3  login           VARCHAR(10) NOT NULL,
4  name            VARCHAR(20) NOT NULL,
5  surname         VARCHAR(20) NOT NULL,
6  address         VARCHAR(40),
7  telephone       VARCHAR(12),
8  maximum_unfinisfed_auctions INTEGER NOT NULL,
9  last_visit      DATETIME,
10 type            VARCHAR(10) NOT NULL);
```

DTO, Example

An implementation of DTO in C# – we have to follow all attributes and their data types of the table `User`.

```
1  public class User
2  {
3      public int Id { get; set; }
4      public String Login{ get; set; }
5      public String Name{ get; set; }
6      public String Surname{ get; set; }
7      public String Address{ get; set; }
8      public String Telephone{ get; set; }
9      public int MaximumUnfinisfedAuctions{ get; set; }
10     public DateTime? LastVisit{ get; set; }
11     public String Type{ get; set; }
12 }
```

DTO, A generator

We can utilize PL/SQL or T-SQL to generate DTO classes.

```
1  CREATE OR ALTER PROCEDURE PrintDtoForTable(@p_table VARCHAR(30))
2  as
3  ...
4  begin
5    set @out = @out + @br + 'public class ' + @p_table;
6    set @out = @out + @br + '{';
7
8    DECLARE c_attr CURSOR LOCAL FOR SELECT cols.name,
9        cols.system_type_id, cols.is_nullable FROM sys.columns cols
10       JOIN sys.tables tbl ON cols.object_id = tbl.object_id
11     WHERE tbl.name=@p_table;
12   OPEN c_attr
13   FETCH NEXT FROM c_attr INTO @a_name,@a_type,@a_nullable
14   ...
```

You can download the implementation from dbedu.cs.vsb.cz, the usage:

```
exec PrintDtoForTable 'User'
```

Relationship 1:1, Example, Tables

A relationship between entities are represented as a reference (a memory pointer), compare to foreign keys in database tables. Let us have two tables:

```
1  CREATE TABLE Account (
2     account_id INT NOT NULL PRIMARY KEY,
3     number INT NOT NULL,
4     code INT NOT NULL
5  );
6
7  CREATE TABLE User (
8     user id INT NOT NULL PRIMARY KEY,
9     fname VARCHAR(20) NOT NULL,
10    lname VARCHAR(20) NOT NULL,
11    email VARCHAR(50) NOT NULL,
12    account INT NOT NULL REFERENCES Account);
```

Relationship 1:1, Example, DTOs

The implementation in Java:

```
1  public class User {
2    private int user_id;
3    private String fname;
4    private String lname;
5    private String email;
6    // we use a reference to a class instance
7    // instead of a foreign key
8    private Account account;
9    .....
10 }
11 public class Account {
12   private int account_id ;
13   private int number;
14   private int code;
15   ...
16 }
```

Relationship 1:N, Example

Let us have two database tables:

```
1  CREATE TABLE Account (
2     account_id INT NOT NULL PRIMARY KEY,
3     number INT NOT NULL,
4     code INT NOT NULL
5     user INT REFERENCES User);
6
7  CREATE TABLE User (
8     user_id INT NOT NULL PRIMARY KEY,
9     fname VARCHAR(20) NOT NULL,
10    lname VARCHAR(20) NOT NULL,
11    email VARCHAR(50) NOT NULL);
```

Relationship 1:N, Variant 1

Variant 1: we follow the database schema.

```
1  public class User {
2    private int user_id;
3    private String fname;
4    private String lname;
5    private String email;
6    ...
7  }
8  public class Account {
9    private int account_id;
10   private int number;
11   private int code;
12   // an object (a reference to the object)
13   // instead of user id is used
14   private User user;
15   ...
16 }
```

Relationship 1:N, Variant 2

Variant 2: a user has a list of accounts.

```
1  public class User {
2    private int user_id;
3    private String fname;
4    private String lname ;
5    private String email;
6    private List<Account> accounts =
7    new ArrayList<Account>();
8    ...
9  }
10 public class Account {
11   private int account_id;
12   private int number;
13   private int code;
14   ...
15 }
```

ORM

- In this course, we will implement a simple ORM (DTO and DAO for each database table).
- Why? We want to optimize the performance of the data layer.
- DAO classes will include methods defined for all functions of the functional analysis.

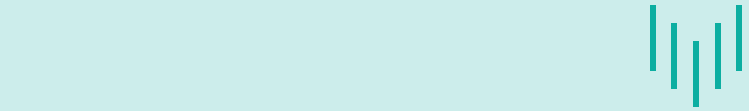
Example, ORM, create.sql

```
CREATE TABLE "User" (
  idUser                    INTEGER NOT NULL PRIMARY KEY IDENTITY,
  login                     VARCHAR(10) NOT NULL,
  name                      VARCHAR(20) NOT NULL,
  surname                   VARCHAR(20) NOT NULL,
  address                   VARCHAR(40),
  telephone                 VARCHAR(12),
  maximum_unfinisfed_auctions INTEGER NOT NULL,
  last_visit                DATETIME,
  type                      VARCHAR(10) NOT NULL);
```

User – DTO

```
1  using System;
2
3  namespace AuctionSystem.ORM
4  {
5   public class User
6   {
7    public int Id { get; set; }
8    public String Login{ get; set; }
9    public String Name{ get; set; }
10   public String Surname{ get; set; }
11   public String Address{ get; set; }
12   public String Telephone{ get; set; }
13   public int MaximumUnfinisfedAuctions{ get; set; }
14   public DateTime? LastVisit{ get; set; }
15   public String Type{ get; set; }
16
17   //Artificial columns (physically not in the database)
18   public String FullName { get
19     { return this.Name + " " + this.Surname; } }
20 }
21 }
```

UserTable – DAO



This class includes the implementation of all functions working with the table `User`.

```
namespace AuctionSystem.ORM.Mssql {
 public class UserTable {
  public static String SQL_SELECT = "SELECT * FROM \"User\"";
  public static String SQL_SELECT_ID = "SELECT * FROM \"User\" WHERE"
    + " idUser=@id";
  public static String SQL_INSERT = "INSERT INTO \"User\" VALUES "
    + "(@login, @name, @surname, @address, @telephone, "
    + "@maximum_unfinisfed_auctions, @last_visit, @type)";
  public static String SQL_DELETE_ID = ...
  public static String SQL_UPDATE = ...
```

Notice:

- We use parametrized operations (to defend the SQL injection).

UserTable.select() 1/3

```
1  protected Collection<User> select(Database pDb = null)
2  {
3    Database db;
4    if (pDb == null) {
5      db = new Database();
6      db.Connect();
7    }
8    else {
9      db = (Database)pDb;
10   }
```

Building the connection to a DBMS takes a few number of seconds, two possible implementations:

- There is a pool of connections, `db.Connect()` means accessing an already open connection from the pool (ASP.NET, J2EE) – **Lines 5–6**.

UserTable.select() 2/3

```
11  protected Collection<User> select(Database pDb = null)
12  {
13    Database db;
14    if (pDb == null) {
15      db = new Database();
16      db.Connect();
17    }
18    else {
19      db = (Database)pDb;
20    }
```

Building the connection to a DBMS takes a few number of seconds, two possible implementations:

- We share one instance of the auxiliary class `Database` (including a connection) among more methods of DAOs – it is pass as a parameter of DAO methods.

UserTable.select() 3/3

```
21      SqlCommand command = db.CreateCommand(SQL_SELECT);
22      SqlDataReader reader = db.Select(command);
23
24      Collection<User> users = Read(reader);
25      reader.Close();
26
27      if (pDb == null)
28      {
29          db.Close();
30      }
31
32      return users;
33  }
```

Line 19: if an instance of `Database` is not passed, we have to close the connection.

UserTable.Read() 1/2


```
1  private static Collection<User> Read(SqlDataReader reader)
2  {
3    Collection<User> users = new Collection<User>();
4
5    while (reader.Read())
6    {
7      int i = -1;
8      User user = new User();
9      user.Id = reader.GetInt32(++i);
10     user.Login = reader.GetString(++i);
11     user.Name = reader.GetString(++i);
12     user.Surname = reader.GetString(++i);
13     user.Address = reader.GetString(++i);
14     user.Telephone = reader.GetString(++i);
15     user.MaximumUnfinisfedAuctions = reader.GetInt32(++i);
```

We use an instance of `SqlDataReader` to read all attribute values, the `Read()` method moves the reading to the next record of the result. All attribute values are stored in the instance of `User`.

UserTable.Read() 2/2

```
16          if (!reader.IsDBNull(++i))
17          {
18              user.LastVisit = reader.GetDateTime(i);
19          }
20          user.Type = reader.GetString(++i);
21
22          users.Add(user);
23      }
24      return users;
25  }
```

When all attribute values are stored in the instance of `User`, this instance is add to the collection of users and it is returned with the method.

UserTable.Delete() 1/2

```
1  protected int delete(int idUser, Database pDb = null)
2  {
3    Database db;
4    if (pDb == null)
5    {
6      db = new Database();
7      db.Connect();
8    }
9    else
10   {
11     db = (Database)pDb;
12   }
```

UserTable.Delete() 2/2

```
13      // SQL_DELETE_ID = "DELETE FROM \"User\" WHERE idUser=@id";
14      SqlCommand command = db.CreateCommand(SQL_DELETE_ID);
15      command.Parameters.AddWithValue("@id", idUser);
16      int ret = db.ExecuteNonQuery(command);
17
18      if (pDb == null)
19      {
20        db.Close();
21      }
22      return ret;
23    }
```

Notice: We use parametrized operations (to defend the SQL injection).

Database 1/4

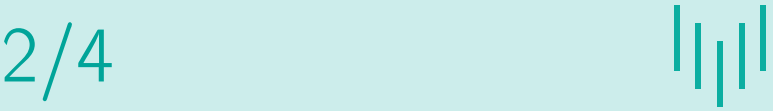
The class `Database` represents an interface to a DBMS.

```
1       public class Database
2       {
3         private SqlConnection Connection { get; set; }
4         private SqlTransaction SqlTransaction { get; set; }
5         public string Language { get; set; }
6
7         public Database()
8         {
9           Connection = new SqlConnection();
10          Language = "en";
11        }
```

Line 3: A connection object is included.

Line 4: A transaction object is included (if it is necessary).

Database - Opening the connection 2/4



```
12  public bool Connect(string conString) {
13    if (Connection.State != System.Data.ConnectionState.Open) {
14      Connection.ConnectionString = conString;
15      Connection.Open();
16    }
17    return true;
18  }
19
20  public bool Connect() {
21    bool ret = true;
22    if (Connection.State != System.Data.ConnectionState.Open) {
23      // connection string is stored in file App.config or Web.confi
24      ret = Connect(ConfigurationManager.ConnectionStrings
25        ["ConnectionStringMsSql"].ConnectionString);
26    }
27    return ret;
28  }
29
30  public void Close() {
31    Connection.Close();
32  }
```

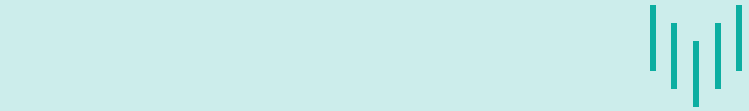
Database – Transaction Support 3/4

```
33  public void BeginTransaction()
34  {
35    SqlTransaction = Connection.BeginTransaction(
36      IsolationLevel.Serializable);
37  }
38
39  public void EndTransaction()
40  {
41    SqlTransaction.Commit();
42    Close();
43  }
44
45  public void Rollback()
46  {
47    SqlTransaction.Rollback();
48  }
```

Database – Update and Query 4/4

```
49      public int ExecuteNonQuery(SqlCommand command)
50      {
51        int rowNumber = 0;
52        try
53        {
54          rowNumber = command.ExecuteNonQuery();
55        }
56        catch (Exception e)
57        {
58          throw e;
59        }
60        return rowNumber;
61      }
62
63      public SqlDataReader Select(SqlCommand command)
64      {
65        SqlDataReader sqlReader = command.ExecuteReader();
66        return sqlReader;
67      }
```

Configuration



`App.config` for a desktop application, `Web.config` for a web application.

```
1  <configuration>
2     ...
3     <connectionStrings>
4     <add name="ConnectionStringMsSql" connectionString="server=..."/>
5     <add name="ConnectionStringOracle"
6     connectionString="Data Source=..."/>
7     </connectionStrings>
8  </configuration>
```

We can define a connection string for various DBMSs, however we must implement extra DAO classes for each of them.

Program.cs – ORM testing 1/3

```
1  namespace AuctionSystem {
2    class Program {
3      static void Main(string[] args) {
4        Database db = new Database();
5        db.Connect();
6        User u = new User();
7        u.Login = "son28";
8        u.Name = "Tonda";
9        u.Surname = "Sobota";
10       u.Address = "Fialová 8, Ostrava, 70833";
11       u.Telephone = "420596784213";
12       u.MaximumUnfinisfedAuctions = 0;
13       u.LastVisit = null;
14       u.Type = "U";
15       UserTableProxy.Insert(u, db);
```

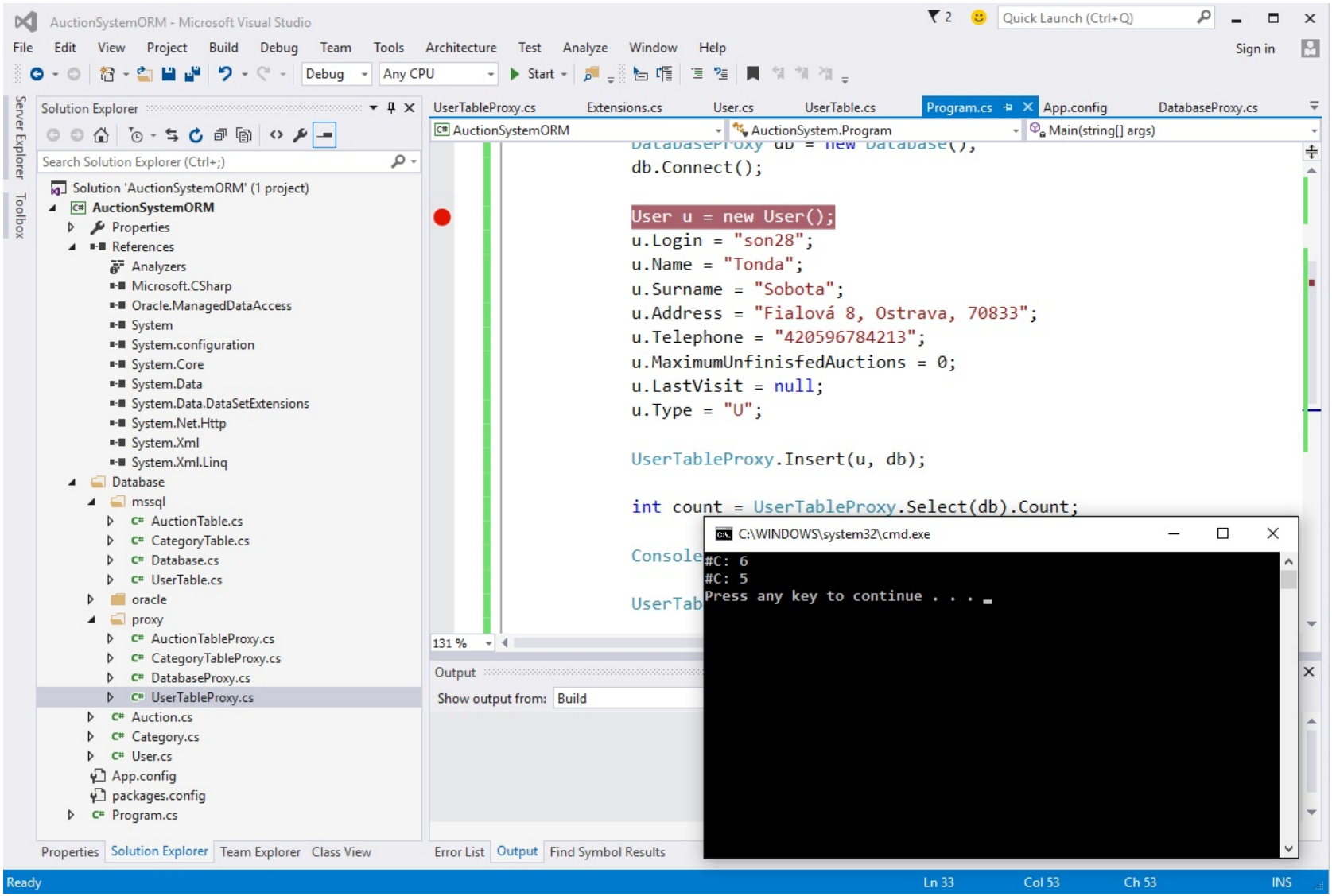
Notice: Line 15: The sample project from dbedu.cs.vsb.cz includes proxy objects working with more DBMSs (Oracle and SQL Server). In your project, you can use only one DBMS.

Program.cs 2/3

```
16          int count1 = UserTable.Select(db).Count;
17          int dltCount = UserTableProxy.Delete(5, db);
18          int count2 = UserTableProxy.Select(db).Count;
19
20          Console.WriteLine("#C: " + count1);
21          Console.WriteLine("#D: " + dltCount);
22          Console.WriteLine("#C: " + count2);
23
24          db.Close();
25      }
26  }
27  }
```

In the semestral project, the test method will include all methods of ORM (i.e. all functions of your functional analysis).

Program.cs 3/3



Semestral Project - ORM, data layer

- To create DTO a DAO classes for all tables of your data analysis.
- To create methods of DAO classes for all functions of your functional analysis.
- To implement an interface of a DBMS – the class `Database`.
- Testing all methods of DAO classes, print the number and name of the function before invocation.
- It is possible to use a sample project from dbedu.cs.vsb.cz

References

- Hibernate: `www.hibernate.org/`
- Java Persistence API (JPA): `https://www.oracle.com/technical-resources/articles/java/jpa.html`
- Entity Framework: `https://docs.microsoft.com/ef/`
- ORMeter: `http://ormeter.net/` – performance testing (but the results are from 2010).