**Komunikace s databázovým systémem**

**Databáze**

**RDBMS** (Relation DataBase Management System)

Není to jen uložiště dat, je to sofistikovaný nástroj, který řeší sousty problémů a zároveň je jednoduchý k použití.

Řeší třeba problém současné editace stejné položky několika uživateli ve stejný okamžik, který by jinak mohl zapříčinit nekonzistenci databáze. RDBMS data v tomto případě zamkne a odemkne až po vykonání zápisu.

Dále umožňuje spojovat několik dotazů do transakcí, kdy se série dotazů vykoná vždy celá nebo vůbec.

Tyto vlastnosti databázového stroje jsou shrnovány zkratkou ACID

ACID - Atomicity (nedělitelnost) Consistency (validita), Isolation (izolace) a Durability (odolnost)

**Atomicity**- transakce se provede celá nebo vůbec. Provádí se jako jedna atomická operace.

= pokud nějaká část operace selže, vrátí se databáze do původního stavu a žádné části transakce nebudou provedeny.

Třeba tohle je důležité v bankovní databázi, když se peníze nepodaří odečíst z prvního účtu tak se nepřipíší na účet druhý

**Consistency**- Stav databáze po dokončení transakce je vždy konzistentní, tedy validní podle všech definovaných pravidel a omezení. Nikdy nenastane situace, že by se databáze nacházela v nekonzistentním stavu.

**Isolation** - Operace jsou izolované a navzájem se neovlivňují. Pokud se sejde v jeden okamžik více dotazů na zápis do stejného řádku, jsou vykonávány postupně, jako ve frontě. FIFO

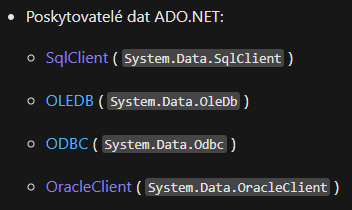
**Durability** - Všechna zapsaná data jsou okamžitě zapsána na trvanlivá úložiště (na pevný disk), v případě výpadku el. energie nebo jiného přerušení provozu RDBMS vše zůstane tak, jak bylo těsně před výpadkem

S databází se komunikuje jazykem SQL

3 základní hotové databáze co umíme

* MSSQL – Microsoft SQL
* MySQL
* Oracle

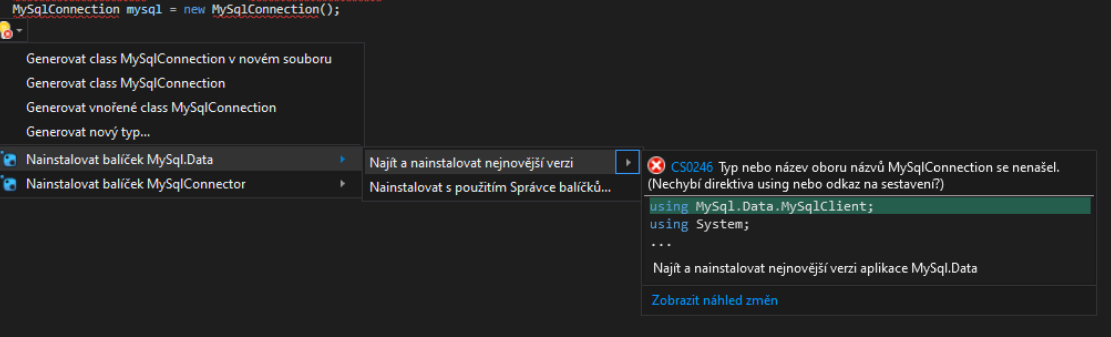
ADO.NET je knihovna, která zastřešuje technologie pro přístup k datům a práci s nimi. Je součástí .NET frameworku.



Visual Studio má nástroje optimalizované pro MSSQL, protože je od Microsoftu stejně jako C# a Visual Studio, nástroje pro Oracle má visual studio taky, protože microsoft s ním počítá, často je určena pro velké podniky.

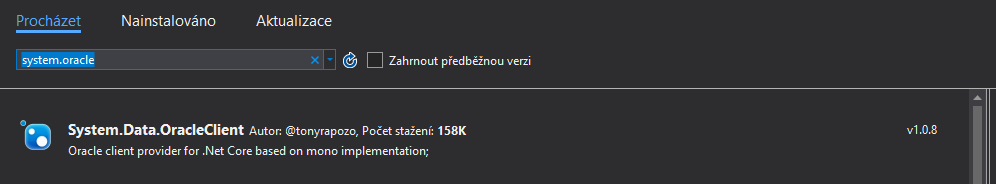
Nástroje pro ostatní databáze jsou zastaralé a jejich vývoj dávno skončil.

MySQL - MySql.Data.MySqlClient

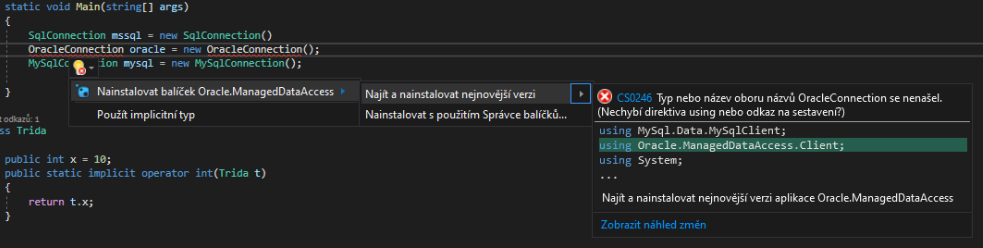


Oracle – System.Data.OracleClient (ADO.NET) nebo Oracle.ManagedDataAccess.Client (ODP.NET)

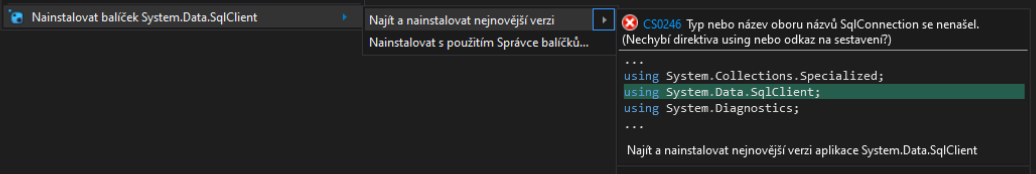
ODP.NET je novější a modernější



(musí se do c# přidat, i když je ADO.NET, tak jako prioritní using je nabídnu ODP.NET)



MSSQL – System.Data.SqlClient



**Rozpor objektového a relačního přístupu**

Relační databáze jsou ověřený způsob jak pracovat s daty. I když existují i databáze plně objektové, tak moc firem je nepoužívá. Např: MongoDB

Revoluce v programování – příchod objektů

= problém s ukládáním dat, jelikož relační databáze objektově nefungují a objekty ukládat neumí.

**Databázový Wrapper**

umožňuje s databází pracovat jako s objektem, nicméně komunikujeme s ní stále v jejím jazyce SQL.

Míchá se objektový a relační kód.

Výhodou je zachování výkonu

### Objektově relační mapování

Z databáze místo pole hodnot se dostanou rovnou objekty a ty na sobě mají metody. SQL jazyk se nepoužívá v programu, tabulky v databázi vidíme jako kolekce objektů, se kterými můžeme pracovat běžnými prostředky jazyka.

Nevýhody: a pozadí dochází k velké degradaci výkonu databáze, SQL dotazy se generují automaticky a jsou často neefektivní. Je těžké naprogramovat

### Objektové databáze

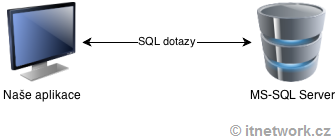
Není moc používaná, drahá atd..

**Připojení do databáze**

**Připojená aplikace**

Když se potřebujeme načíst, vkládat, měnit nebo mazat data v reálném čase

DataReader, Command, Connection obsahují SQL příkazy a dostávají výsledky, jako kdyby se SQL příkazy psaly v databázovém programu.



**Odpojená aplikace – DataSet**

Jedná se o Databázový Wrapper

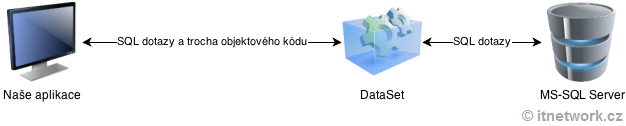
V operační paměti je DataSet, který v sobě má data se kterýma aplikace pracuje.

Občas se DataSet sesynchronituje s databází.

DataSet v sobě  obsahuje tabulky (DataTable), tabulka řádky(DataRow) a řádek sloupce (DataColumn).

Tabulka je objekt. Můžou se do ní řádky přidávat a upravovat je bez psaní SQL kódu.

Když chceme spustit na databázi nějaký příkaz, použijeme DataAdapter, pomocí kterého si naplníme DataSet daty. Příkazy již musíme psát v jazyce SQL dané databáze.



**LINQ To SQL**

Jedná se o objektově relační mapování

LINQ to SQL poskytuje kompletní objektovou abstrakci nad databází

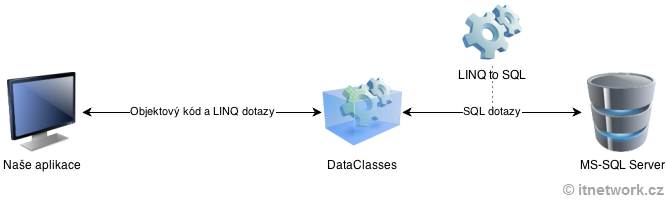
Databázi v programu vidíme jako List Objektů

Nevíme o tabulkách, od příkazů dostáváme rovnout objekty

Vše se děje na pozadí automaticky

Pracujeme s DataClasses, objektrová struktura databáze, obsahuje třídy pro jednotlivé tabulky, sloupce tabulky jsou vlastnosti daných tříd.

pouze s DataClasses, ta potom na pozadí pomocí LINQ to SQL komunikuje s databází a provádí za nás SQL dotazy.



Nevýhoda, horší výkon

### Entity Framework

Pokročilejší LINQ TO SQL,

Umí vazbu M:N

Ještě horší výkon

**Připojení z C# do DB**

Abychom se mohli k DB připojit, potřebujeme tzv. ConnectionString. To je řetězec, který obsahuje údaje potřebné k připojení k databázi. Typicky je zde název databáze a případně heslo.

Píše se do závorek deklarace konekce nebo jako její vlastnost u všech 3.

**MSSQL**

DataSource = adresa db serveru

InitialCatalog = název db

Password =heslo

UserID = username

**MYSQL**

Server/Host/Data Source/Data Source/Address = adresa db serveru

Database /Initial Catalog = název db

Password/pwd =heslo

Username/User Id/Uid = username

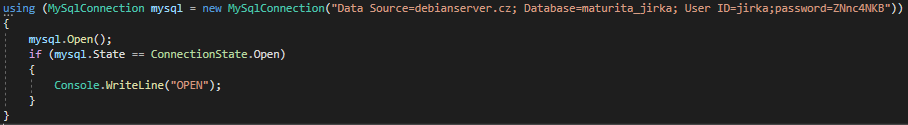


Nebo 

Nebo

SQLConnectionStringBuilder ten má jako vlastnosti Username,Password atd…

Je dobrým zvykem všechny databázové objekty tvořit v bloku using, aby došlo k rychlému zavření nepotřebných spojení.



ConnectionState je v System.Data

Většina tříd je stejná jen podle databáze jinak zaříná jejich název OracleCommand, SqlCommand a MySqlCommand

## Databázové příkazy

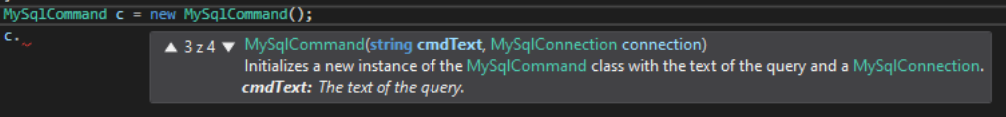
Třída SqlCommand nebo MySqlCommand, k dispozici je taky třída builder jako u connection stringu.

Píše se do ní příkazy v jazyce SQL.

Důležité vlastnosti jsou

Connection = instance connection (v jaké db se má command provést)

CommandText = příkaz v jazyce SQL



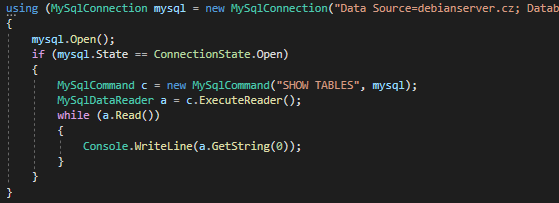
Často se zapomíná na connection

Metody na spuštení

**ExecuteReader()** - Tuto první využijeme tehdy, pokud SQL dotaz vrací množinu dat (většina příkazů SELECT, např. dotaz na jedno nebo více slovíček).

**ExecuteScalar()** - Druhá metoda nalezne své uplatnění, pokud SQL dotaz vrací jedinou hodnotu (typicky SELECT s agregační funkcí, např. počet všech slovíček).

**ExecuteNonQuery()** - Poslední metoda se používá ke spouštění dotazů, které nevracejí žádnou hodnotu (příkazy INSERT, UPDATE, DELETE, např. vložení slovíčka). Její návratová hodnota nese informaci o počtu "ovlivněných" řádků.

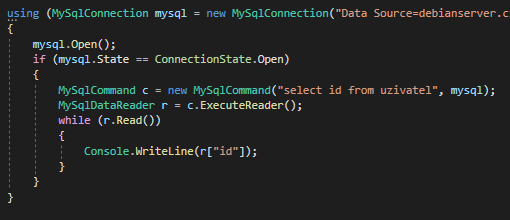


ExecuteReader() a ExecuteScalar() vrací MySqlDataReader (pro mysql)

Ta umožňuje procházet výslednou množinu dat po jednotlivých záznamech.

Metoda Read(), ta se dává do while cyklu (dokud bude mít řádky ke čtění), protože nevíme kolik záznamů je v tabulce

Pro přístup k hodnotě na řádku se používá **r.GetInt32(index)** (převede na požadovaný datový typ) nebo **r[“nazev sloupce“]** nebo **r[index]**

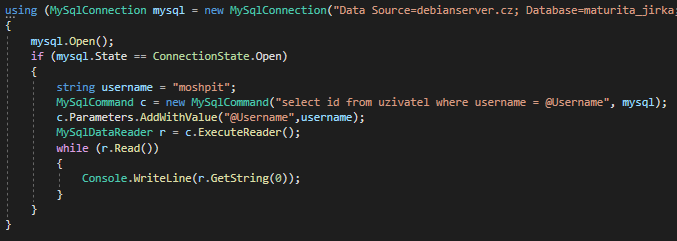


Zadání textu do commandu, není dobrý dělat řetězec pomocí string+string

Ani string + nejakejObject.Jmeno.ToString()

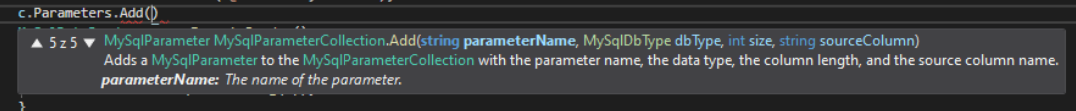
Protože když třeba to nejakejObject.Jmeno nastavím na “DROP DATABASE neco“ tak autor programu umožnil smazat svým programem svoji databázi.

To se nazývá SQL injection



Ochrana pomocí Parameters

AddWithValue(“@proměná v sql commandu“,zdroj naší promění co chce vložit, nebo její obsah)



AddValue je použit třeba u DataGridView ve Forms

DB si sama parametry ošetří, takže se tam škodolibý příkaz nedostane

Když v máme v DB třeba proměnou where, tak by kolidovala s podmínkovým příkazem, proto se proměný píšou do hranatých závorek



V MySQL je to Grave accent ` 🡨 tohle místo hranatých závorek

**DataSet a DataAdapter**

DataSet umožňuje uchovávat a upravovat velké množství dat v lokální paměti bez stálého připojení ke zdroji dat, proto je odpojená aplikace.

DataAdapter slouží jako most mezi DataSet a SQL Server pro načítání a ukládání dat.

Obsahuje kolekci komponent DataTable (ve vlastnosti Tables), které reprezentují jednotlivé databázové tabulky (DataTable).

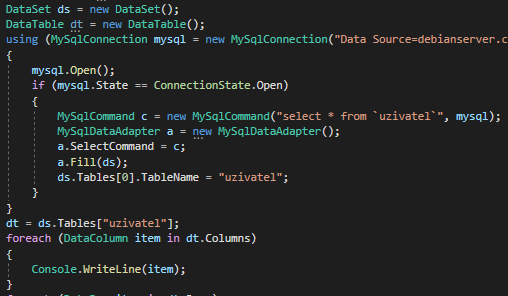
Navíc může obsahovat informace o vztazích mezi těmito tabulkami.

Často je DataAdapter pouze pro jednu tabulku.

Poskytuje tento most podle mapování Fill , který změní data v, DataSet aby odpovídala datům ve zdroji dat, a Update který změní data ve zdroji dat tak, aby se shodovala s daty v, a to DataSet pomocí příslušných příkazů jazyka Transact-SQL pro zdroj dat. Aktualizace se provádí po jednotlivých řádcích. U každého vloženého, změněného a odstraněného řádku Update Metoda určuje typ změny, která byla provedena v této metodě ( Insert , Update nebo Delete )

V DataSet paměti, která je mezipaměť dat načtená ze zdroje dat, je hlavní součástí architektury ADO.NET. DataSetObsahuje kolekci DataTable objektů, které lze vzájemně propojit s DataRelation objekty. Integritu dat můžete také vymáhat DataSet pomocí UniqueConstraint ForeignKeyConstraint objektů

Vypsání názvů sloupců v tabulce uživatel v DataTable

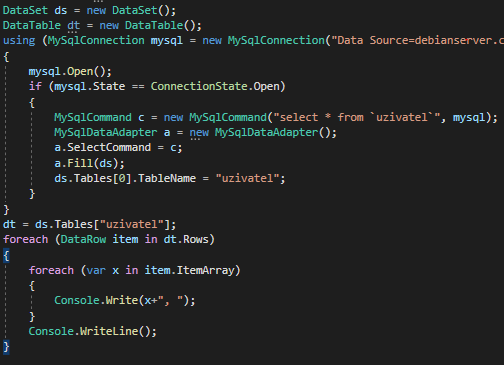


Jméno tabulky typu DataTable v DataSetu se dá přidat v metodě Adapteru Fill()

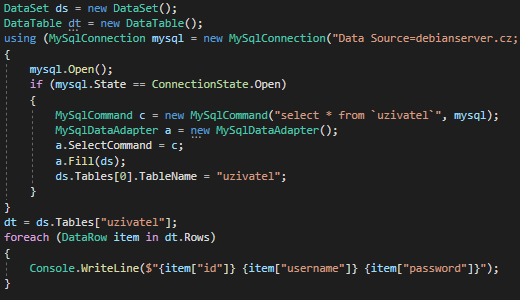


Když se tabulka nepojmenuje tak má jméno “Table“, je důležité pojmenovávat tabulky, jinak se všechno ukládá do jedný

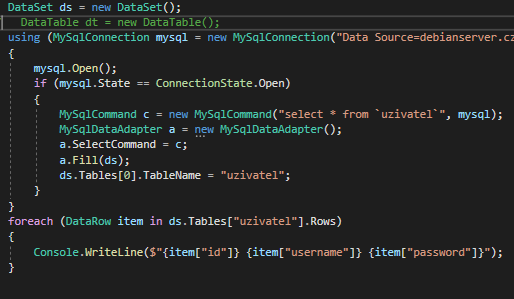
Vypsání celých řádků v tabulce uživatel v DataTable



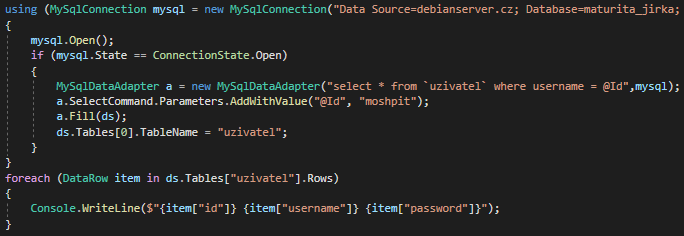
Vypsání určitých sloupců v tabulce uživatel v DataTable



Vypsání určitých sloupců v tabulce uživatel bez DataTable



Adapter do svého konstruktoru dokáže vytvořit třídu Command ze String a Connection co mu dáme

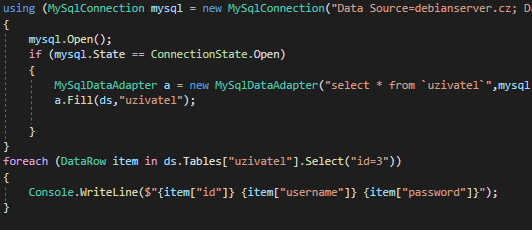


Tato metoda se nejčastěji volá s jedním parametrem - filtr (podmínka)

nebo se dvěma parametry (filtr, řazení dat).

Tento způsob je výhodný právě v konzolové aplikaci.

Select z tabulky, kterou máme v DataSet

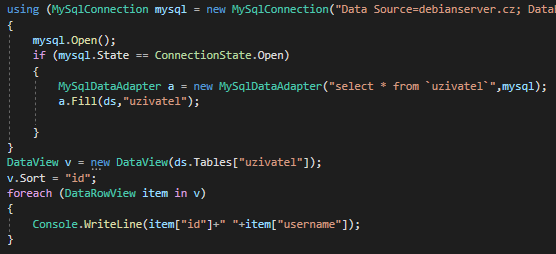


DataView je třída, která umožňuje různé pohledy na data v komponentě DataTable.

Mezi její základní vlastnosti patří RowFilter, Sort a RowStateFilter.

Základní pohled na tabulku získáme z vlastnosti DefaultView třídy DataTable.

Tento způsob se více využívá ve formulářových aplikacích, neboť DataView můžeme použít přímo jako datový zdroj (DataSource) pro komponentu DataGridView.

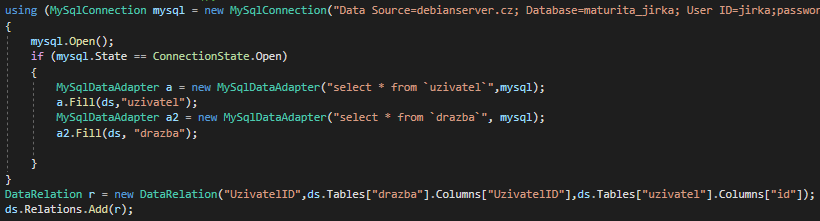


Spojení tabulke v DataSetu

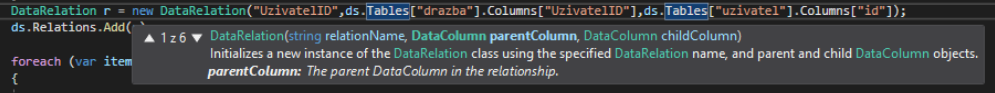
Tato relace se při přenosu dat do DataSet ztratí.

V C# je relace reprezentována třídou DataRelation

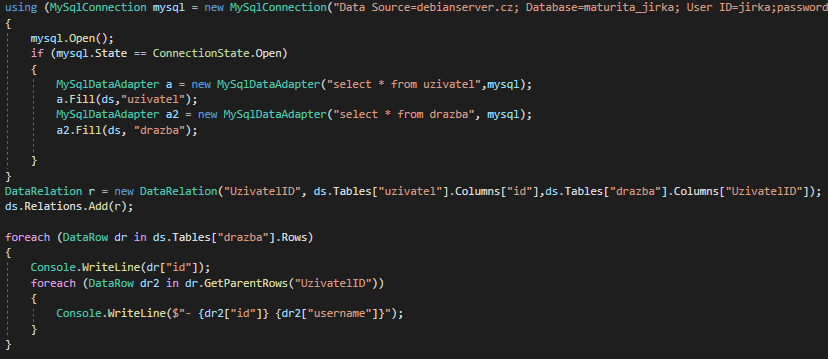
v DataSet jsou všechny relace uloženy v kolekci Relations.



childColumn je tabulka ve které je foreign key



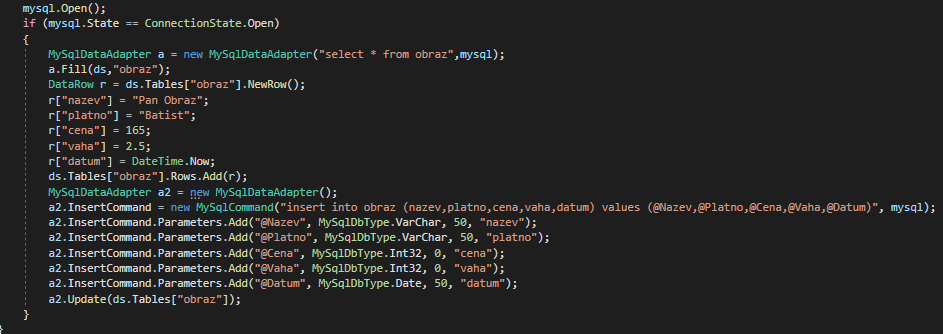
Relaci přidáváme do DataSet jen tehdy, pokud ji plánujeme nějak využít. Typickým příkladem je využití metod GetChildRows a GetParentRow třídy DataRow, které s pomocí relace dokáží získat odpovídající data z vázané tabulky.



Vložení z DataSetu do DB

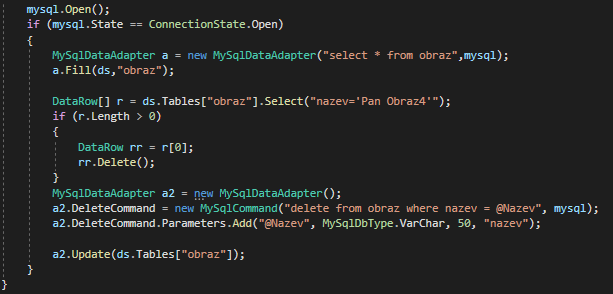
Vytřit DataRow v DataSetu. Vložit jí do DataSetu. Vytvořit Adapter, napsat InsertCommand.

Update() je používá pro insert,delete a update;



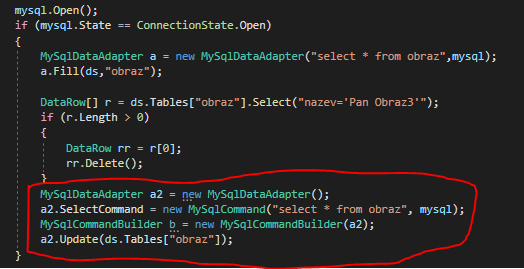
Mazání

Najít v DataSetu row kterou chcem smazat. Vytvořit Delete command. A Update



Psaní commandů update, delete a insert by bylo moc dlouhé

MySqlCommandBuilder je umí vygenerovat, potřebuje k nim jenom select command, aby věděl jak se daná tabulka v DB jmenuje.



**Datové typy v C# na DB typy**

