Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Obor: Informační technologie ve vzdělávání

Katedra informatiky

Editor ERA modelů

Bakalářská práce

Jan Dolan

Vedoucí práce

RNDr. Hana Havelková

Český Krumlov 2011

Prohlášení:

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum: Podpis:

Poděkování

Touto cestou bych chtěl poděkovat mé vedoucí bakalářské práce RNDr. Haně Havelkové za optimistický přístup a trpělivost při konzultacích. Dále bych chtěl poděkovat Bc. Veronice Kempské a celé své rodině za jejich velkou duševní podporu.

Obsah

1 Úvod

V současné době se informační technologie objevují i v těch nejběžnějších lidských činnostech. Každý rozsáhlejší informační systém bývá zpravidla řízen databází, tento přístup je označován jako database-driven. Databáze řídí celý systém, stará se o informace něm uložené, řídí konkurenční přístup k datům a zároveň zajišťuje datovou integritu a bezpečnost. Díky rozmachu informačních technologií se výuka informatiky stává jednou z klíčových aspektů školství. Na vytváření a údržbu databázových systémů je potřeba úzce specializované odborníky, kterých je na trhu práce potřeba čím dál tím více.

Ve výuce databází je pro studenty klíčové pochopit, jak konceptuálně navrhnout takový systém. Tento návrh se provádí za pomoci entitně-relačních diagramů a napomáhá definovat abstraktní pohled na určitou problematiku v systému. Na základě takových diagramů lze mnohem efektivněji navrhnout tabulkový databázový model a zefektivnit tak proces vývoje aplikací.

Pro kvalitní výuku problematiky entitně-relačních návrhů je vhodné podpořit ji programovým prostředím pro návrh entitně-relačních diagramů. Vytvoření takového prostředí je náplní této bakalářské práce. Takovéto prostředí by mělo být schopné podpořit návrh entitně-relačních diagramů, tedy definovat entity, relace a vztahy mezi nimi. Toto prostředí bude realizováno ve formě síťové aplikace, bez potřeby instalace, přístupné z internetu v internetovém prohlížeči, naprogramované pomocí technologií ASP.NET a Silverlight. Aplikace bude obsahovat i správu uživatelských účtů. V aplikaci bude možno ukládat a načítat své projekty a z nich generovat kód jazyka SQL definující strukturu relační databáze v MS-SQL serveru 2008.

Existuje několik prostředí použitelných pro školní prostředí. Můžeme využít běžného editoru vektorové grafiky, který je však pro potřeby modeláře entitně-relačních diagramů zbytečně složitý a není schopen vygenerovat kód SQL. Dalším východiskem může být prostředí Visual Paradigm, které ve volně šiřitelné verzi Community edition negeneruje kód a jeho notace se spíše přibližuje UML. Dalším prostředím je ER modeller, který se používá k výuce na pedagogické fakultě Jihočeské univerzity.

Tato bakalářská práce popisuje nově vzniklé prostředí pro modelování entitně-relačních diagramů a slouží jako metodická příručka k softwaru. Měla by napomoci těm, kteří se rozhodnou podpořit svou výuku touto aplikací i těm kteří se rozhodnou toto prostředí rozšířit. Práce je rozdělena do pěti kapitol. Následující kapitola se zabývá samotnou problematikou entitně-relačního modelování, napomáhá pochopit jak tyto diagramy navrhovat, k čemu slouží, a jak poté definují vztahy mezi tabulkami v databázi. Kapitola třetí popisuje, jak entitně-relační diagram převést na diagram relační, který definuje databázi v dnešních databázových systémech. Čtvrtá kapitola se zabývá realizací

aplikace a její metodikou, nasazením systému a jeho nároky. Kapitola pátá popisuje prostředí aplikace a to z pohledu jak uživatelského tak i administrátorského. Poslední, šestá kapitola se zabývá zhodnocením úspěšnosti vytvoření aplikace za použití průzkumu mezi studenty na základě porovnání s aplikací ER modeller.

2 Entitně-relační modelování

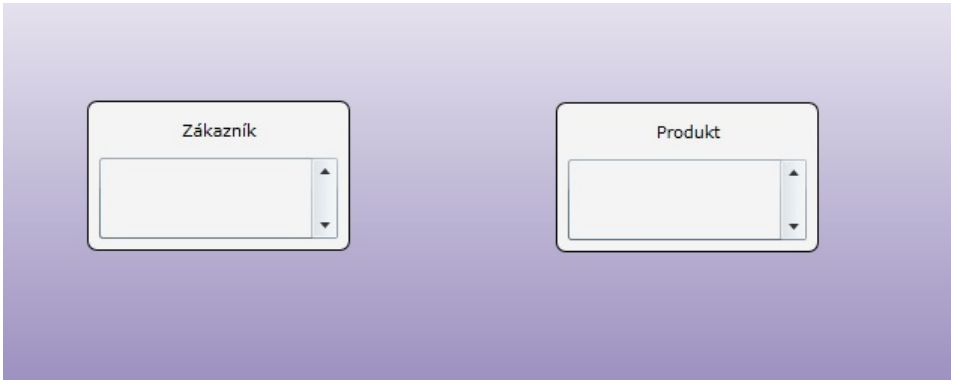
Entitně-relační modelování je proces návrhu databáze označovaný jako shora dolů. Samotné modelování probíhá tak, že nejprve určíme ta data, která jsou pro systém nejdůležitější (tzv. entity), poté definujeme vztahy mezi daty (tzv. relace, či relationship) a nakonec určujeme detailně definovaná data a vztahy (např. atributy dat, či specifikace kardinality vztahů). Modelování tedy probíhá postupně od abstraktnějšího pohledu na data ke konkrétnějšímu. V následujících podkapitolách postupně rozeberu jednotlivé části entitně-relačního modelu.

2.1 Entita

Jedním ze základních kamenů entitně-relačního modelování je entita, což napovídá sám název. Entitu je možné definovat jako určitý shluk informací v systému, které spolu navzájem souvisí a společně tvoří nějaký objekt. Entita představuje abstraktní pohled na daný objekt. Tento objekt se pak v systému může vyskytovat vícekrát. Jako příklad poslouží například firma, která vyrábí zahradní nábytek. Tato firma vyrábí nějaké produkty a má nějaké zákazníky. Jak produkt, tak i zákazník jsou objekty, které se vyskytují v systému firmy a jsou tedy entitami. Produkt je abstraktní pojem, entita bude tedy nazvána „produkt" a bude obsahovat konkrétní produkty, tedy například židle, stůl, altán a podobně. Stejně tak je to i se zákazníkem kdy zákazník je abstrakce nad určitou množinou osob, které u firmy nakoupili. Název entity je zpravidla určen podstatným jménem. [3]

Každá entita je doplněna informacemi, které blíže specifikují její vlastnosti. Každou vlastnost entity označujeme jako atribut. Databáze zpravidla obsahuje mnoho entit a každá entita obsahuje mnoho atributů. O atributech se podrobněji zmíním v podkapitole 2.3.

Grafická reprezentace entity v diagramu je zpravidla obdélník. Viz následující diagram.



Obrázek 1: Znázornění entity

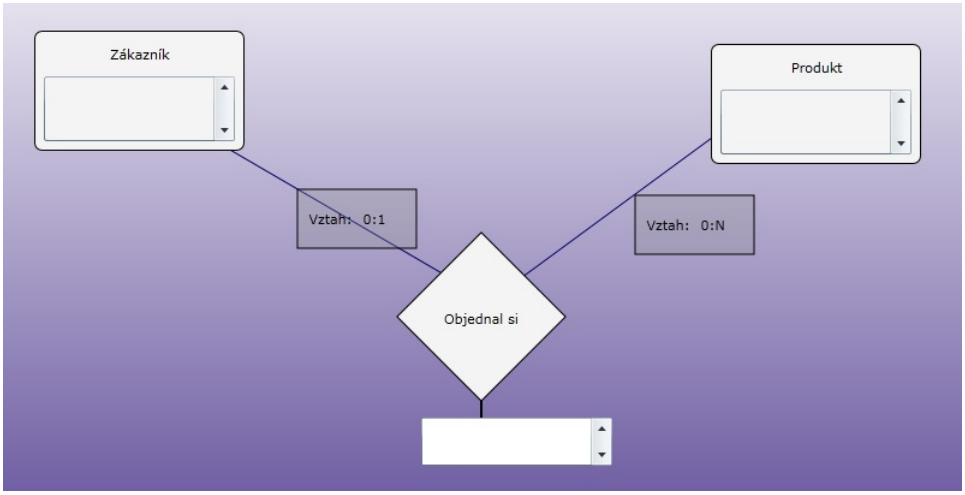
2.2 Relace

Relace je propojení určitých entit podle jejich vzájemného vztahu. V databázi se může vyskytovat spoustu relací, přičemž relace může spojovat dvě a více entit. Relace je obvykle identifikována slovesem.

Jako příklad relace můžeme uvést například následující vztah. U firmy se zahradním nábytkem si zákazník objedná určité produkty. Takový vztah pak mezi entitou Zákazník a entitou Produkt vytváří relaci „Objednal si". Pokud složíme názvy entit a relací, pak nám vyjde vztah „Zákazník si objednal produkt". [3]

Každá relace se vyznačuje také svým stupněm relace. Ten určuje kolik entit se v relaci vyskytuje. Nejčastěji narazíme na relaci binární, ale obecně se může vyskytnout jakákoli n-ární relace.

Graficky se relace v entitně-relačním diagramu značí kosočtvercem. Viz následující diagram.



Obrázek 2: Znázornění relace

2.3 Atribut

Atributy blíže specifikují entitu či relaci. Každý atribut obsahuje nějaká konkrétní data relevantní pro danou entitu. Například entita zákazník bude obsahovat atributy jméno, příjmení, bydliště a další.

Atribut může nabývat určitých speciálních vlastností.

Povinný atribut (mandatory) udává, že v každém řádku entity musí být data atributu vyplněny. V aplikaci označen černým kruhem.

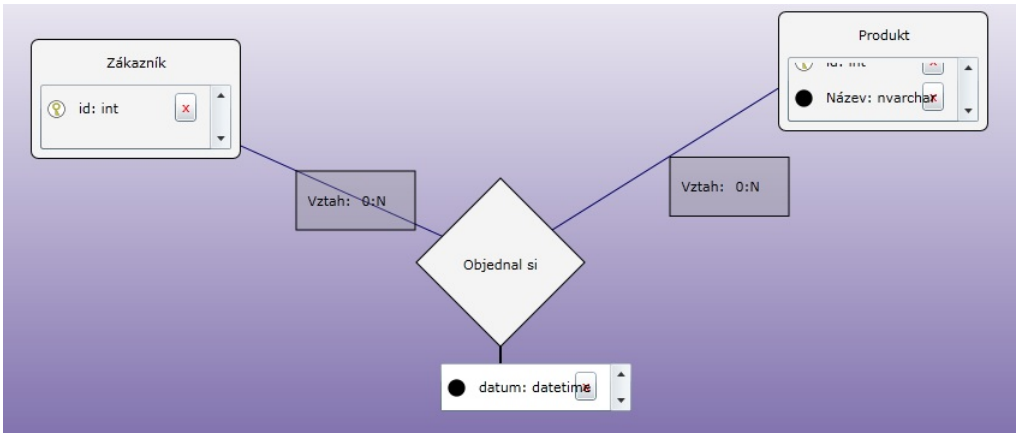
Nepovinný atribut (non-mandatory) udává, že atribut nemusí být v každém řádku vyplněn. V aplikaci označen prázdným kruhem.

Unikátní atribut (unique) definuje takový atribut, v jehož množině dat se žádné nevyskytuje dvakrát. Každá hodnota atributu musí být mezi všemi hodnotami atributu v dané entitě unikátní. V aplikaci označen písmenem U v kružnici.

Primární klíč (primary key) je unikátní atribut, který zvolíme pro jedinečnou identifikaci příslušné entity. Každá entita by měla mít svůj primární klíč, jehož pomocí se pak mohou definovat relace vůči jiným entitám. Osobu například unikátně identifikuje její rodné číslo. V aplikaci označen ikonou klíče v kružnici. [3] [1]

Atributy se mohou vyskytovat i v relacích, obvykle je to nějaká další doplňující informace k relaci mezi dvěma řádky dat. Například pokud chceme k objednávce přidat ještě datum objednání. [3]

Atribut se obvykle graficky znázorňuje pomocí elipsy. U velkých tabulek však toto zobrazení není příliš přehledné. Proto jsem v aplikaci raději zvolil posuvný seznam, který je vypsán uvnitř obdélníku relace. Viz obrázek 3.



Obrázek 3: Znázornění atributu

2.4 Kardinalita relací

U každé relace je nutno ještě definovat kardinalitu takového vztahu. Existují tři vztahy mezi entitami, které se mohou vyskytnout a to vztah 1:1, 1:N a M:N. Každý z nich má určitý význam a rozpadá se na dvojici vztahů. Ty pak definují vztah mezi entitou a relací. Tento vztah by měl vycházet z logického vztahu v reálném světě, Vztahy viz následující tabulka. [3]

Pro lepší pochopení uvedu ke každému vztahu příklad.

[1] Pozn. autora: Osobu sice můžeme snadno unikátně určit pomocí rodného čísla, tento postup se však z bezpečnostních důvodů nepoužívá. Je vhodnější vytvořit si vlastní identifikátor jako zákaznické číslo apod.

Vztah	Rozklad vztahu	Význam
1:1	0:1 ku 0:1	Vztah 1:1 se vyskytne v případě, že k jednomu záznamu jedné entity existuje nanejvýš jeden záznam entity druhé
1:N	0:1 ku 0:N	Vztah 1:N se vyskytuje, pokud k jednomu záznamu v první entitě se vyskytuje více záznamů v entitě druhé, ovšem jeden záznam v entitě druhé k sobě váže nanejvýš jeden záznam z první entity.
M:N	0:N ku 0:M	Vztah M:N se vyskytuje v případě, že k záznamu v první entitě se vyskytuje více záznamů v druhé entitě a zároveň k záznamu ve druhé entitě se vyskytuje více záznamů v entitě první.

Tabulka 1: Kardinalita mezi entitami

Vztah 1:1 se vyskytuje poměrně zřídka, jelikož je možné tento vztah zjednodušit tak, že sloučíme obě entity do jedné, někdy je ale tento vztah nutný, buď pro zachování přehlednosti, nebo pokud potřebujeme napojit novou entitu na starší a není možné z technických důvodů starou entitu předefinovat. Takový vztah se tedy například může vyskytnout mezi entitami Zákazník a Uživatelský účet. Jeden zákazník vlastní pouze jeden uživatelský účet a zároveň jeden uživatelský účet patří pouze jednomu zákazníkovi.

Vztah 1:N se vyskytuje v databázích velmi často. Například mezi entitami Zákazník a Faktura by měl existovat vztah 1:N, protože jeden zákazník vlastní několik faktur, ale jedna faktura patří jen jednomu zákazníkovi.

Vztah M:N se vyskytuje například mezi entitami Faktura a Produkt. Jedna faktura může obsahovat více produktů a zároveň jeden produkt se může nacházet ve více fakturách.

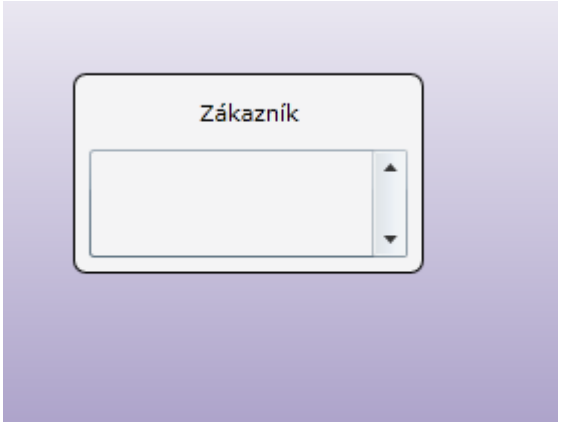
Každý vztah má svůj význam při převodu entitně-relačního modelu na relační model, tato problematika je rozebrána v kapitole 3.

3 Převod mezi entitně-relačním a relačním schématem

Převod z entitně-relačnímu diagramu přímo do kódu SQL je velkým usnadněním práce s databázemi. Tento převod je možný, ale převod nikdy není úplně stoprocentní. Entitně-relační diagram určuje sice abstraktní pohled na data a vztahy mezi nimi, ale databáze dokáže definovat mnohem sofistikovanější funkce, jako například definice odvozených atributů [2] či pokročilé testování integrity dat [3]. Další omezení této konverze je použitý databázový server. Ač existují určité konvence jazyka SQL, každý databázový server si syntaxi trochu přizpůsobil. Viz vzniklé odvozeniny jazyka jako T-SQL, PL/SQL a další. Převod do jazyka SQL musí být tedy přizpůsoben jazyku daného serveru. V aplikaci je převod přizpůsoben jazyku T-SQL, který se využívá v prostředí databázového serveru Microsoft SQL server.

3.1 Převod entit

Každá entita je v relačním schématu dána právě jednou tabulkou. Tato tabulka nese jméno entity. Viz přiložený obrázek a jeho reprezentace v jazyce SQL.



Obrázek 4: Převod entity na SQL

```
USE [Master]

GO

CREATE DATABASE [Testovaci_databaze]

GO
```

[2] Odvozené atributy se spočítají automaticky z jiných atributů, jako např. věk z data narození apod.
[3] Některé atributy musí být například sudé apod.

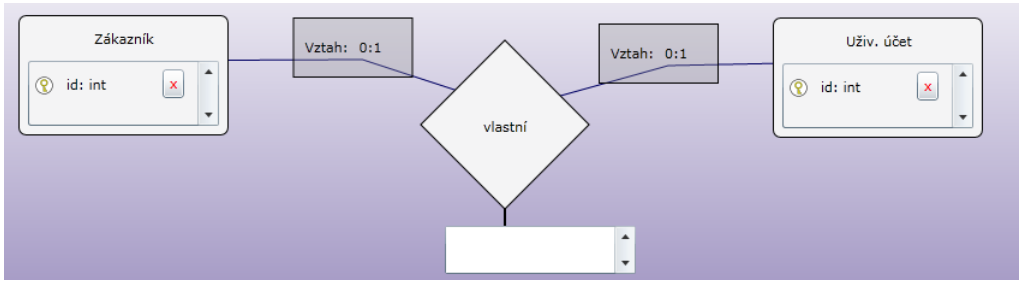
```
USE [Testovaci_databaze]

CREATE TABLE [dbo].[Zákazník] ( )
```

3.2 Převod relací

Převod relace závisí na její kardinalitě vůči entitám, které spojuje. Relace je však vždy reprezentována cizím klíčem, který vzniká z důsledku závislosti entit na sobě. Podmínkou vytvoření relace je existence primárního klíče v některé z entit. Ve které entitě musí klíč být závisí opět na kardinalitě vztahu, ovšem všeobecně se doporučuje, aby každá entita obsahovala primární klíč. [3]

Dalším důsledkem vytvoření relace, respektive jejího cizího klíče je vznik referenční integrity dat. Pokud bude existovat v atributu cizího klíče odkaz na hodnotu primárního klíče, řádek s touto hodnotou nemůže být odstraněn a nemůže být změněna hodnota primárního klíče. Odkaz v cizím klíči by totiž odkazoval na data, která v databázi neexistují.

V případě vztahu 1:1 vzniká nový atribut v jedné z entit (nezáleží ve které). Tento atribut se označuje jako cizí klíč. Ten odkazuje na primární klíč z druhé entity. Aby šlo o vztah 1:1, musí být atribut cizího klíče označen jako unikátní, tedy žádná hodnota se v tomto atributu nesmí opakovat, každý řádek má svou unikátní.

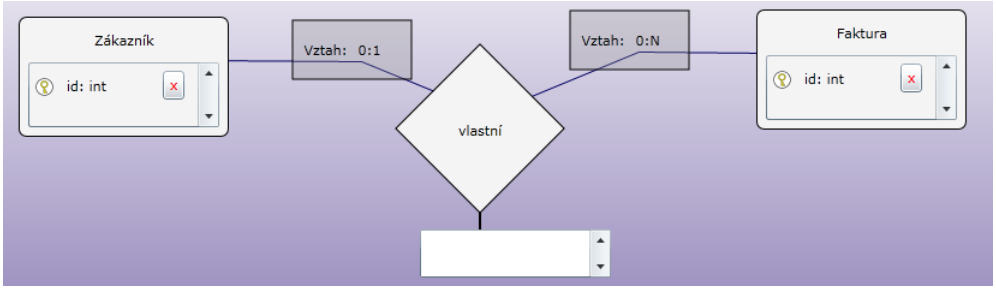


Obrázek 5: Vztah 1:1

```
USE [Master]

GO

CREATE DATABASE [Test serializace]
```

```
GO

USE [Test serializace]

CREATE TABLE [dbo].[Zákazník]

(

 [Uživ. účetID] [int]UNIQUE, [id] [int] IDENTITY(1,1) NOT NULL,


 CONSTRAINT [PK_Zákazník] PRIMARY KEY CLUSTERED ([id] ASC)

)

CREATE TABLE [dbo].[Uživ. účet]

(

 [id] [int] IDENTITY(1,1) NOT NULL,

 CONSTRAINT [PK_Uživ. účet] PRIMARY KEY CLUSTERED ([id] ASC)

)

ALTER TABLE [dbo].[Uživ. účet] WITH CHECK ADD CONSTRAINT
[FK_Zákazník_Uživ. účet] FOREIGN KEY([Uživ. účetID]) REFERENCES
[dbo].[Zákazník] ([id])
```

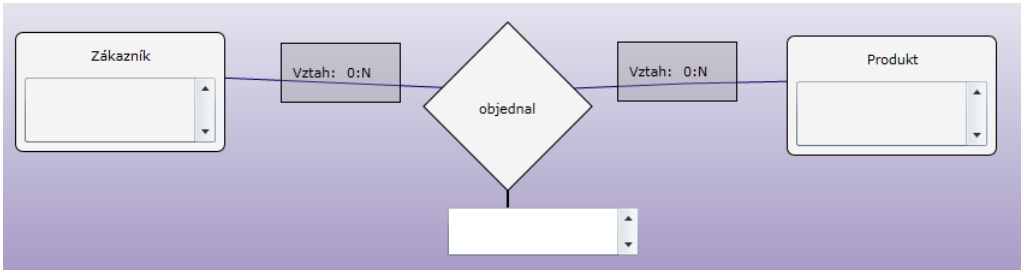
Vztah 1:N je obdobou vztahu 1:1, liší se však v unikátnosti atributu cizího klíče. Daný atribut není unikátní, ve vztahu 1:N se entita na straně N (v naší notaci na straně spojení 0:N) mohou opakovat reference na stejný řádek druhé entity. Navíc u tohoto vztahu záleží, u které entity bude primární a u které cizí klíč. Jak již předchozí část napovídá, u entity, která se objevuje ve části 0:1, se vyskytne primární klíč a entita ve vztahu 0:N bude obsahovat cizí klíč. Opět vznikne referenční integrita jako v předcházejícím případě, pouze může nastat situace kdy budou integritní pravidla porušena několikrát, v případě že více řádků bude odkazovat na ten odstraňovaný.



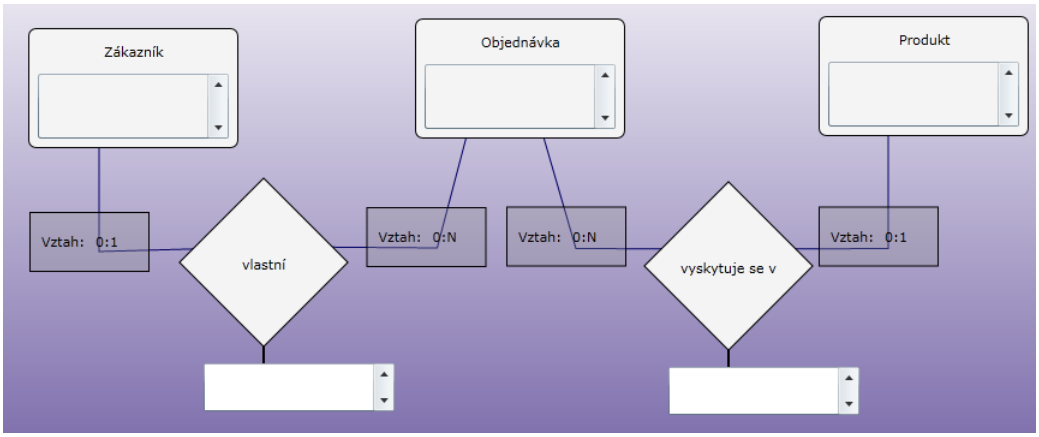
Obrázek 6: Vztah 1:N

```
USE [Master]
GO
CREATE DATABASE [Test serializace]
GO
USE [Test serializace]
CREATE TABLE [dbo].[Zákazník]
(
 [id] [int] IDENTITY(1,1) NOT NULL,
 CONSTRAINT [PK_Zákazník] PRIMARY KEY CLUSTERED ( [id] ASC)
)
CREATE TABLE [dbo].[Faktura]
(
 [ZákazníkID] [int] NULL, [id] [int] IDENTITY(1,1) NOT NULL,
 CONSTRAINT [PK_Faktura] PRIMARY KEY CLUSTERED ( [id] ASC)
)
ALTER TABLE [dbo].[Zákazník] WITH CHECK ADD CONSTRAINT
[FK_Faktura_Zákazník] FOREIGN KEY([ZákazníkID]) REFERENCES
[dbo].[Faktura] ([id])
```

Vztah M:N není možné do tabulek entit nijak definovat. Musí tedy nastat rozklad na jednodušší vztahy. Tento proces se nazývá dekompozice. Vytvoříme novou tabulku, čímž ze vztahu M:N (0:N a 0:N) v relaci Zákazník ku Produkt vznikne vztah 1:N N:1 (respektive 0:1 0:N 0:N 0:1) ve vztahu Zákazník ku Objednávka ku Produkt. Viz následující schémata.



Obrázek 7: Vztah M:N



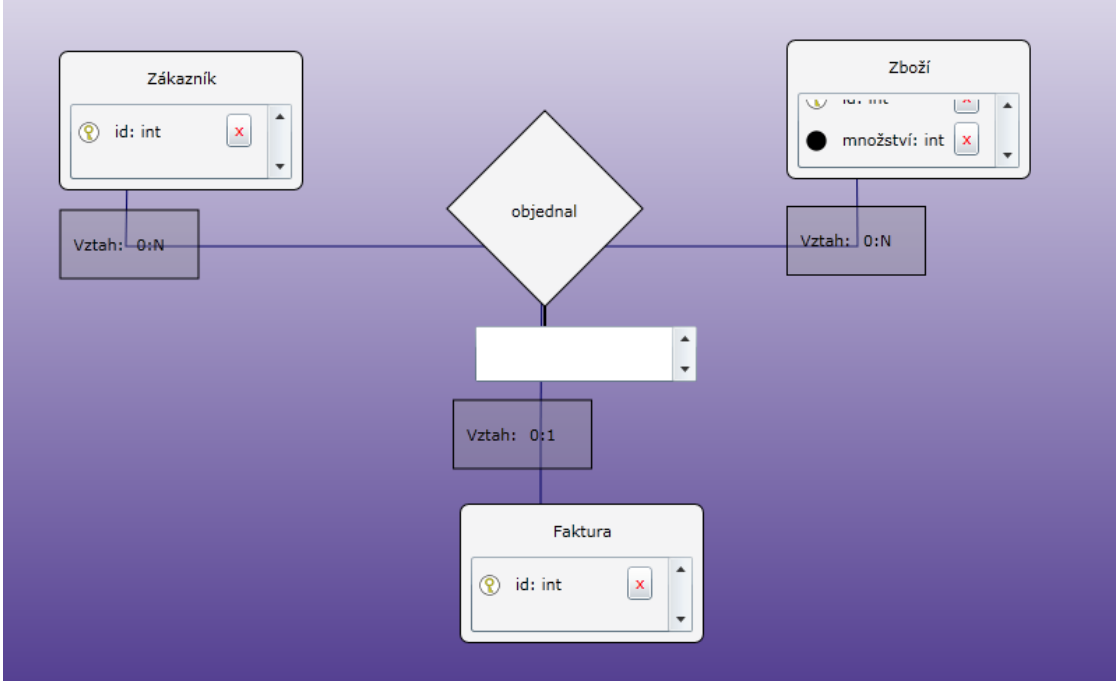
Obrázek 8: Dekompozice vztahu M:N

```
USE [Master]

GO

CREATE DATABASE [Testovací_db]

GO

USE [Testovací_db]

CREATE TABLE [dbo].[Zákazník]

(

 [id] [int] IDENTITY(1,1) NOT NULL, CONSTRAINT [PK_Zákazník]
PRIMARY KEY CLUSTERED ( [id] ASC)

)
```

```
CREATE TABLE [dbo].[Produkt]

(

 [id] [int] IDENTITY(1,1) NOT NULL, CONSTRAINT [PK_Produkt]
PRIMARY KEY CLUSTERED ( [id] ASC)

)

CREATE TABLE [dbo].[Zákazník_Produkt]

(

 [id] [int] IDENTITY(1,1) NOT NULL,

 [Zákazník_ID] [int] NOT NULL,

 [Produkt_ID] [int] NOT NULL,

 CONSTRAINT [PK_Zákazník_Produkt] PRIMARY KEY CLUSTERED ( [id]
ASC )

)



ALTER TABLE [dbo].[Zákazník_Produkt] WITH CHECK ADD CONSTRAINT

[FK_Zákazník_Zákazník_Produkt] FOREIGN KEY([Zákazník_ID])
REFERENCES [dbo].[Zákazník] ([id])

ALTER TABLE [dbo].[Zákazník_Produkt] WITH CHECK ADD CONSTRAINT

[FK_Produkt_Zákazník_Produkt] FOREIGN KEY([Produkt_ID])
REFERENCES [dbo].[Produkt] ([id])
```

3.3 Převod ternárních a vícenásobných relací

Ternární relace je taková, která sdružuje 3 entity. Její stupeň vztahu je právě 3. Obecně může existovat n-násobná relace. Převod do SQL pak vypadá následovně.



Obrázek 9: Ternární relace

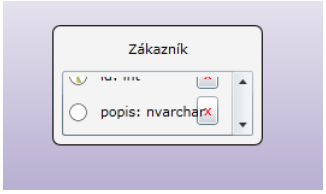
Je patrné že bylo pouze nutné vytvořit více atributů a cizích klíčů, nikoli tabulek.

```
USE [Master]
GO
CREATE DATABASE [Testovací_DB] GO
USE [Testovací_DB]
CREATE TABLE [dbo].[Zákazník]
(
 [id] [int] IDENTITY(1,1) NOT NULL,
 CONSTRAINT [PK_Zákazník] PRIMARY KEY CLUSTERED ( [id] ASC)
)
CREATE TABLE [dbo].[Faktura]
(
 [id] [int] IDENTITY(1,1) NOT NULL,
 CONSTRAINT [PK_Faktura] PRIMARY KEY CLUSTERED ( [id] ASC)
```

```
)
CREATE TABLE [dbo].[Zboží]
(
 [id] [int] IDENTITY(1,1) NOT NULL, [množství] [int] NOT NULL,
 CONSTRAINT [PK_Zboží] PRIMARY KEY CLUSTERED ( [id] ASC)
)
CREATE TABLE [dbo].[Zákazník_Zboží]
(
 [id] [int] IDENTITY(1,1) NOT NULL,
 [Zboží_ID] [int] NOT NULL,
 [Faktura_ID] [int] UNIQUE NOT NULL,
 [Zákazník_ID] [int] NOT NULL,
 CONSTRAINT [PK_Zákazník_Zboží] PRIMARY KEY CLUSTERED ( [id]
ASC )
)
ALTER TABLE [dbo].[Zákazník_Zboží] WITH CHECK ADD CONSTRAINT
[FK_Zboží_Zákazník_Zboží]
FOREIGN KEY([Zboží_ID]) REFERENCES [dbo].[Zboží] ([id])
ALTER TABLE [dbo].[Zákazník_Zboží] WITH CHECK ADD CONSTRAINT
[FK_Faktura_Zákazník_Zboží]
FOREIGN KEY([Faktura_ID]) REFERENCES [dbo].[Faktura] ([id])
ALTER TABLE [dbo].[Zákazník_Zboží] WITH CHECK ADD CONSTRAINT
[FK_Zákazník_Zákazník_Zboží]
FOREIGN KEY([Zákazník_ID]) REFERENCES [dbo].[Zákazník] ([id])
```

3.4 Převod atributů

Každý atribut je definován svým jménem, datovým typem a speciální vlastností, jak bylo popsáno výše. Příklad převodu entity s atributy viz následující schéma.



Obrázek 10: Převod atributu na SQL

```
USE [Master]

GO

CREATE DATABASE [Testovaci_databaze]

GO

USE [Testovaci_databaze]

CREATE TABLE [dbo].[Zákazník] ( [id] [int] IDENTITY(1,1) NOT
NULL, [popis] [nvarchar] NULL, CONSTRAINT [PK_Zákazník]

PRIMARY KEY CLUSTERED ( [id] ASC))
```

3.5 Posloupnost příkazů SQL

Pro správné a bezchybné zpracování je potřeba, aby příkazy SQL splňovaly následující pořadí:

1. Nasměrování kontextu dotazu na hlavní databázi master. (USE [Master])
2. Vytvoření databáze (CREATE DATABASE [Název databáze])
3. Vytvoření tabulek (CREATE TABLE...)
4. Vytvoření cizích klíčů (ALTER TABLE [dbo].[Název tabulky] WITH CHECK CONSTRAINT...)

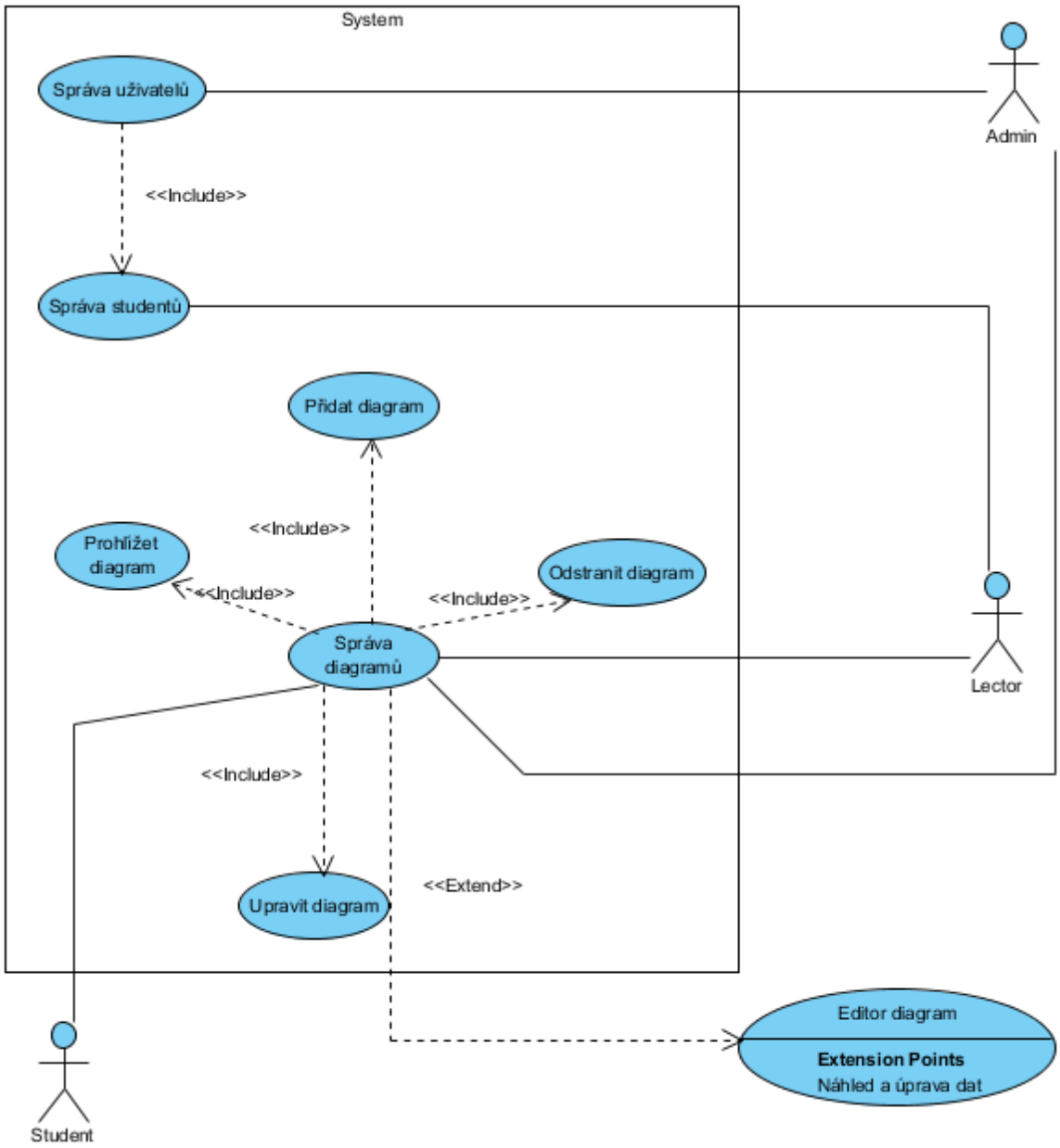
4 Metodika a realizace aplikace

4.1 Návrh UML

Každá dnešní moderní aplikace se neobejde bez návrhu pomocí modelovacího jazyka UML. Nejinak jsem postupoval v případě této aplikace. K vytvoření diagramů bylo použito nástroje Visual Paradigm, který umožňuje modelování na profesionální úrovni.

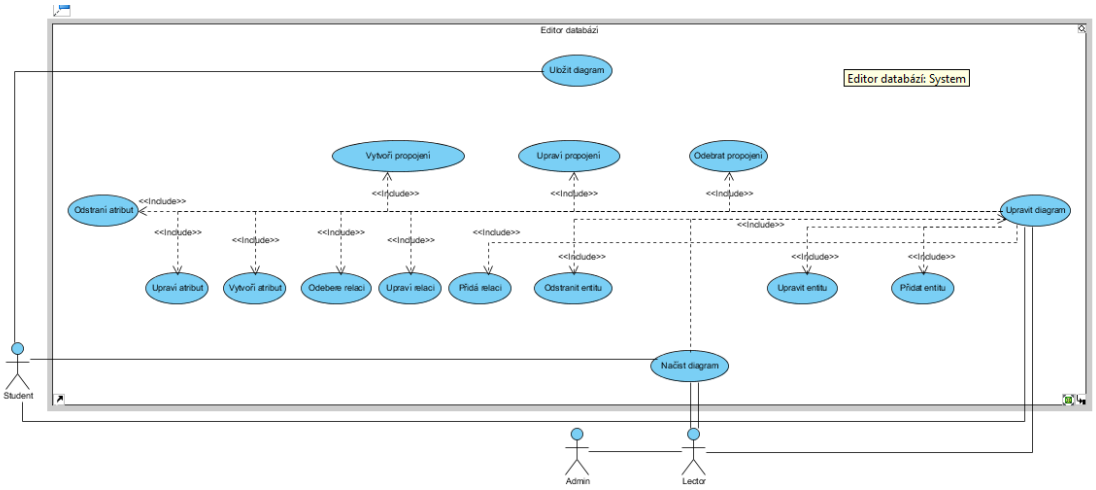
4.1.1 Návrh případů užití

Pro návrh jednotlivých možností aplikace je vhodné použít Use Case diagramy. Následující obrázek znázorňuje případy užití aplikace.



Obrázek 11: Use Case diagram aplikace

Z diagramu je patrné, že aplikace sestává z několika částí a to správy uživatelů, správy studentů a správy diagramů. Dále definuje tři role existující v systému. Role Admin reprezentuje správce systému, ten má možnost vstupovat kamkoli, včetně správy uživatelů systému. Role Lector charakterizuje správce veškerých Entitně-Relačních diagramů, veškeré diagramy může zároveň procházet. Pro optimalizaci k výuce může uživatel s právy Lector specifikovat studenty, kteří mají k aplikaci přístup. Poslední rolí v systému je Student, tato role umožňuje pouze přistupovat k editoru Entitně-Relačních diagramů. Editace diagramů je patrnější z následujícího diagramu. Ten popisuje komponentu, která upravuje diagramy.



Obrázek 12: Use Case diagram komponenty pro editaci E-R diagramů.

Z diagramu je patrné, že komponenta pro editaci diagramů je nejrobustnější částí celé aplikace. Co se týče přístupových práv jednotlivých rolí, pak práva administrátora a lektora se shodují. Obě role smí přistupovat k jakémukoli diagramu, bez ohledu na vlastnictví. Student smí přistupovat pouze k diagramům, které sám vytvořil. Tato komponenta se skládá ze správy diagramů (načítání, úprava). Úprava diagramů pak zahrnuje vytváření, úpravu a mazání entit, relací a atributů. Dále pak definuje propojení mezi entitou a relací.

4.2 Rozdělení na projekty

Z Use Case diagramů je možno aplikaci rozštěpit na menší celky, podle funkčnosti. Na každé funkce se totiž (nejen dle zadání) hodí různé technologie. Aplikaci jsem tedy rozdělil na tři části.

Pro realizaci zdrojového kódu bylo použito nástroje Microsoft Visual Web Developer 2010.

4.2.1 Aplikace ASP.NET

Tato část zajišťuje správu uživatelských účtů a zpřístupňuje komponentu pro úpravu diagramů. Vzhledem k volbě webové aplikace, využívající technologii .NET jsem zvolil aplikaci ASP.NET a to formou webových formulářů, tato komponenta není příliš velká a není potřeba rozdělovat práci více programátorům. Bylo by tedy zbytečné volit verzi využívající návrhový vzor MVC.

Správa uživatelských účtů

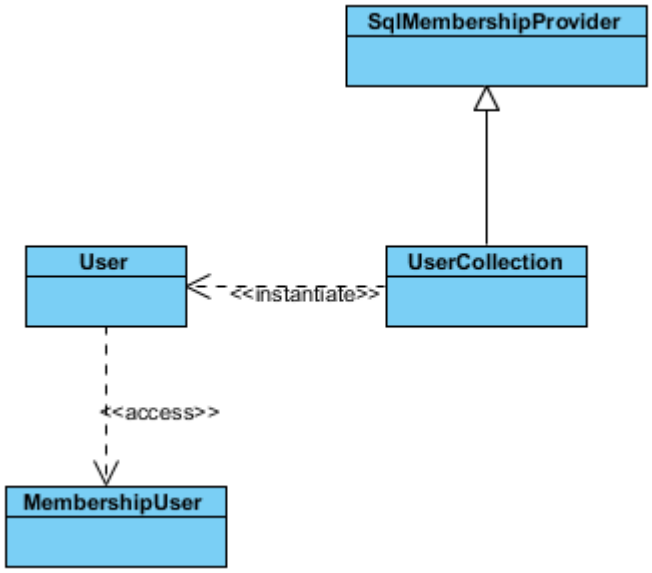
Správu uživatelských účtů umožňují formuláře ve složce AccountAdmin. Zde je

tedy možno jednotlivě přidávat, upravovat a mazat uživatele. Vzhledem k potřebě vkládání mnoha studentů je umožněno přidávání uživatelů dávkou. Dávkový soubor je možno asynchronně vložit, po úspěšném uploadu se zobrazí náhled přidaných uživatelů. Ten slouží pro kontrolu.

Dávkový soubor má pevně definovaný formát.

- Soubor musí být ve formátu CSV a splňovat následující podmínky. Hodnoty musí být odděleny středníkem.
- Data musí být v pořadí: Číslo studenta, Příjmení, Jméno, Login.

Pro přihlašování jsem použil již hotové rozhraní .NET, bylo by nesmyslné vytvářet nějaký vlastní systém přihlašování, vzhledem k tomu, že toto rozhraní je odladěno, dobře zabezpečeno a nadále rozvíjeno přímo v technologii .NET. Bylo nutno upravit rozhraní MembershipProvider pro potřeby aplikace, protože objekt uživatele obsahuje data, která jej charakterizují. Model přihlašování je definován podle následujícího diagramu.



Obrázek 13: Rozšíření přihlašovacího rozhraní .NET

Zpřístupnění komponenty pro tvorbu E-R diagramů

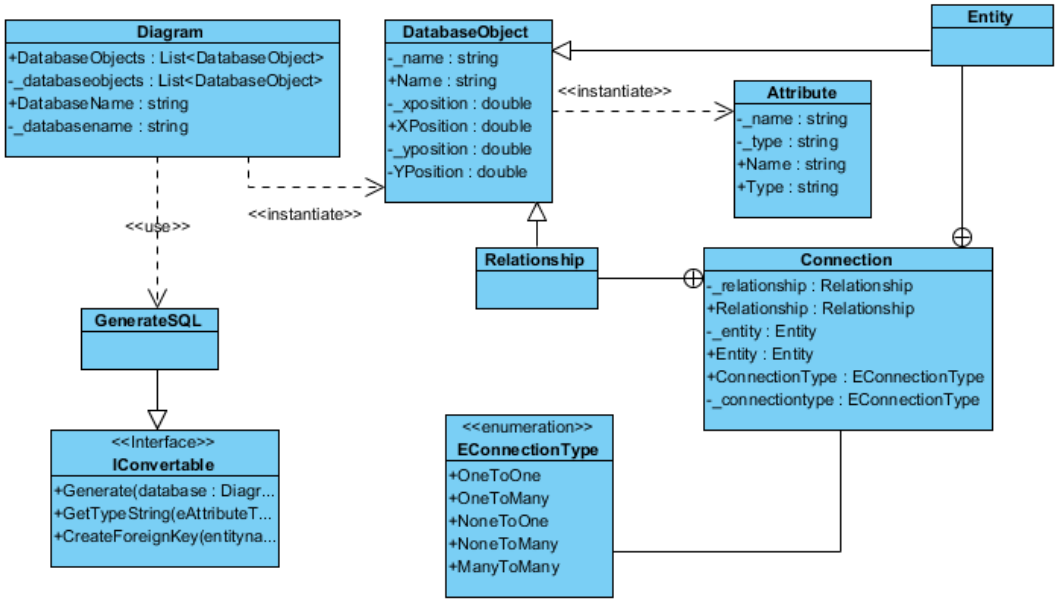
Webová část projektu musí plnit úlohu zpřístupnění aplikace pro editaci Entitně-Relačních diagramů, protože je tato součást systému vytvořena za pomoci technologie Silverlight 4.0. Pro tento účel slouží formulář Application.aspx ve složce DBToolController. Samotná aplikace je reprezentována souborem DBToolController.xap, což je spouštěcí soubor editoru diagramů.

Webové služby

Webové služby definované ve složce Handlers/WebServices zprostředkovávají informaci o přihlášeném uživateli editoru diagramů. Webovou službu je nutno definovat, protože editor, díky technologii Silverlight běží na straně klienta a nezatěžuje server. Zároveň však nemá přístup k serverovým proměnným, jako je právě jméno přihlášeného uživatele.

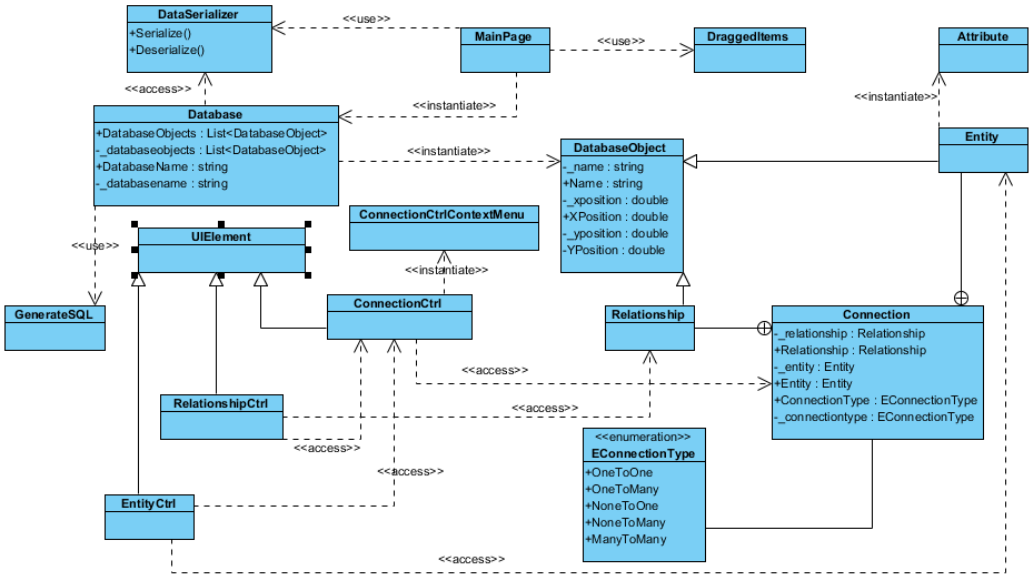
4.2.2 Aplikace Silverlight

Tato součást systému obsahuje i definici objektové struktury diagramu. Tu je potřeba dobře promyslet, protože bez kvalitního objektově orientovaného řešení se struktura diagramu stane velmi nepřehlednou, těžko serializovatelnou a stěží zobrazitelnou. Objektově orientovaná struktura diagramu je vyobrazena na následujícím diagramu.



Obrázek 14: Objektový pohled na E-R diagram

Dále se na tuto strukturu váže grafická nadstavba, která zprostředkovává celou vizualizaci diagramu, funkce drag&drop a uživatelskou editaci diagramu. Každá vizuální část je odvozena od třídy UIElement, která zprostředkovává grafické rozhraní. Objektovou strukturu a provázání s objektovým pohledem na data znázorňuje následující diagram.



Obrázek 15: Provázání objektového pohledu na diagram a grafického rozhraní

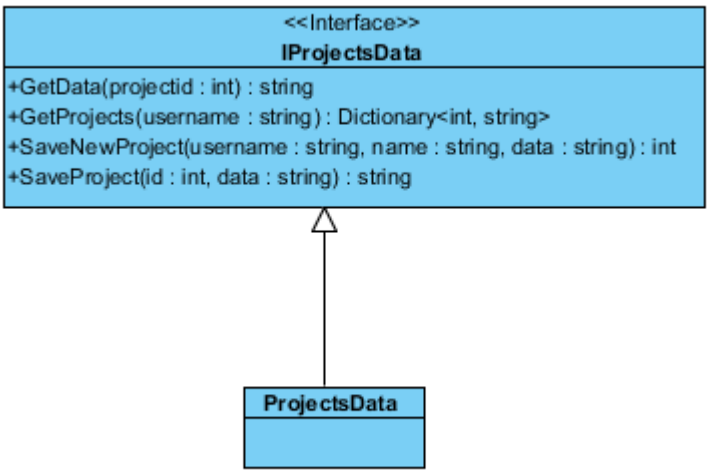
4.2.3 Webové datové služby

Další nedílnou součástí projektu jsou webové datové služby. Jak již bylo řečeno, technologie Silverlight je spuštěna u klienta, nikterak nezatěžuje server, ale nemá přímý přístup k serverovým prostředkům a databázi, která je na serveru provozována. Pro komunikaci s databází je tedy nutno definovat webovou službu a potřebné metody.

V dnešní době se využívá jmenného prostoru WCF (Windows communication foundation), který obsahuje nadstavbu nad původním .NET remotingem a staršími webovými službami. Ten je schopen komunikovat pomocí SOAP, které je automatizováno.

Webovou službu reprezentuje projekt DBToolDataService. Ten obsahuje rozhraní IProjectData, které definuje metody, které se využívají pro komunikaci. Toto rozhraní je označeno jako [ServiceContract], tedy definice pro webovou službu, každá metoda je pak označena jako [OperationContract], nebo-li operace, kterou služba vykonává (metoda přístupná pomocí webové služby).

Třída ProjectsData.svc.cs je odvozena od rozhraní IProjectData a definuje metody. Zde se již nachází přímo kód ke komunikaci s databází. Serializace dat je však prováděna opět na straně klienta. Díky tomu se opět nezatíží server. Pro usnadnění zobrazuje následující diagram objektový návrh webové služby.

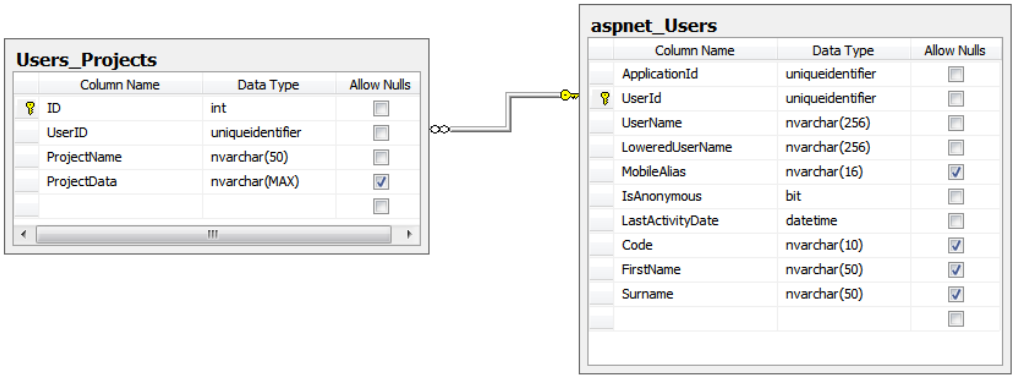


Obrázek 16: Objektový návrh webové služby

4.3 Návrh databáze a zpracování dat

4.3.1 Databázový diagram

Jak již bylo řečeno, každá moderní aplikace se neobejde bez databáze a tato není výjimkou. Pro realizaci databáze jsem pro dobrou kompatibilitu zvolil MS-SQL Server 2008. Databáze obsahuje standardní tabulky databáze pro skladování informací o uživatelích (všechny s prefixem aspnet) a tabulku Users_Projects. Ta obsahuje samotná serializovaná data projektů, které se k uživatelům vážou. Vztah provázání původní .NET databáze a tabulky Users_Projects je vyobrazena na následujícím diagramu.



Obrázek 17: Databázový diagram - napojení tabulek aspnet_Users a Users_Projects

Jak je tedy zřejmé, data jsou uložena v atributu ProjectData tabulky Users_Projects, tato data jsou uložena ve formátu nvarchar(MAX), tedy v běžném textovém řetězci. Důvod použití tohoto datového typu je zřejmý z následující podkapitoly.

4.3.2 Serializace dat

Serializace objektové struktury se může stát velice problematickou. Samozřejmě je možnost data serializovat v bajtovém proudu, což však není rozumné provádět v prostředí webu, kde se vyskytuje více technologií. XML má několik dalších výhod, mezi které patří například čitelnost člověkem, přehledná struktura a další. [4]

Technologie .NET nám od verze 3.0 přináší snazší, přehlednější a jednoduše definovatelný typ serializace. Namísto staršího označování tříd jako [Serializable] je nyní možno lépe specifikovat, jak se má objektová struktura serializovat. Tato změna přišla spolu s Windows Communication Foundation, která zároveň přinesla nový typ webových služeb. V objektové struktuře definujeme každou třídu, která se má serializovat jako [DataContract], každou proměnnou nebo vlastnost, která se má serializovat jako [DataMember]. Pokud nastane případ, že na objekt existuje odkaz z několika míst, je třeba ještě definovat takovýto vztah pomocí klauzule [DataContract(IsReference = true)]. [5]

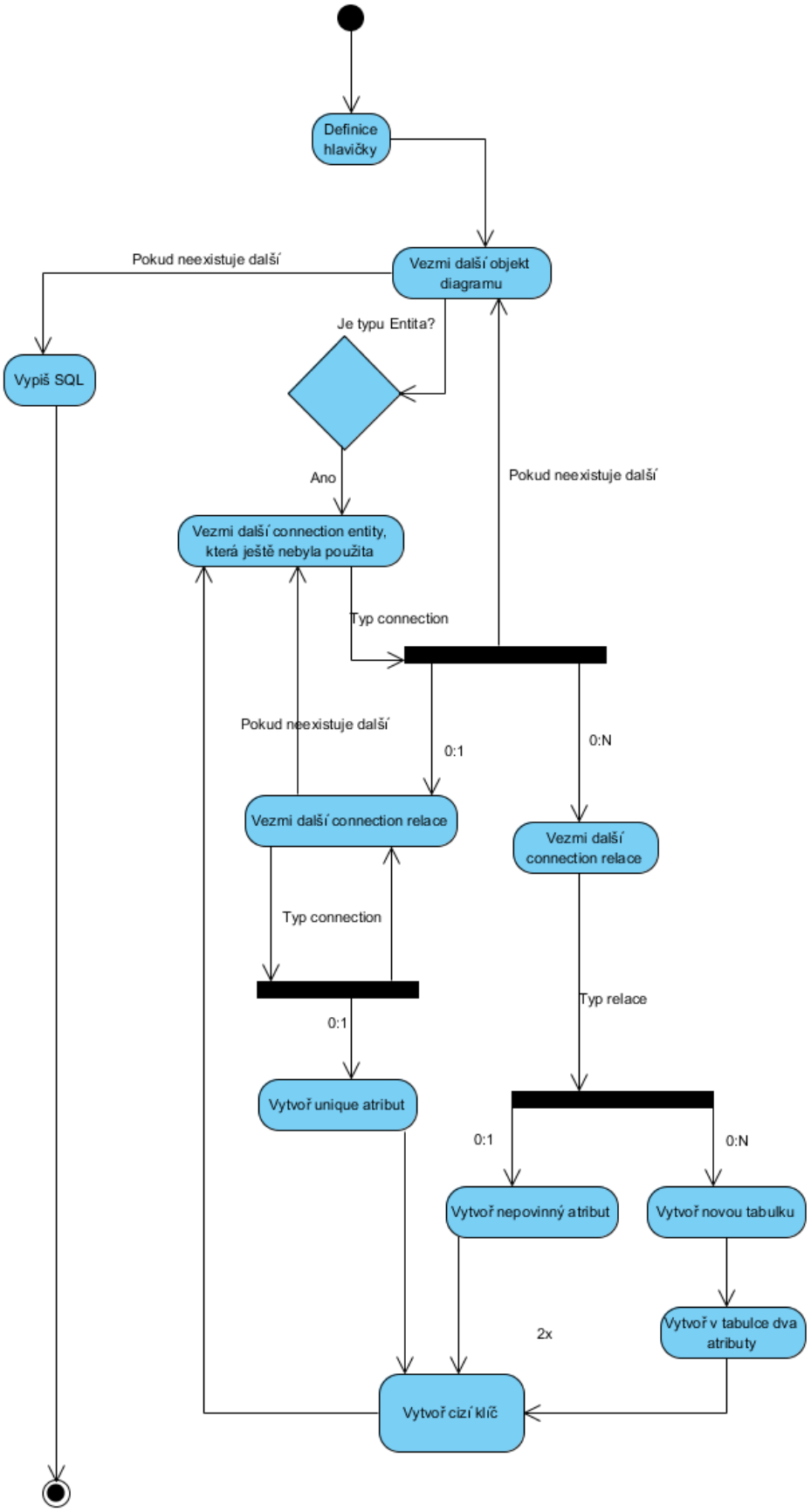
Sama technologie pak již podporuje převod objektové struktury do formátu XML, který se využívá pro komunikaci s webovou službou pomocí protokolu SOAP. Proces serializace a deserializace XML je v aplikaci reprezentován generickými metodami `static string Serialize<T>` a `static T Deserialize<T>` ze třídy DataSerializer. [6]

4.4 Algoritmizace převodu E-R diagramu do jazyka SQL

Převod mezi Entitně-Relačním schématem a jazykem SQL je definován rozhraním IConvertable. Každá třída, která definuje takový převod, by měla být od tohoto rozhraní odvozena. Jak již bylo řečeno, převod mezi E-R diagramem a jazykem SQL musí být přizpůsoben danému databázovému stroji. V případě této aplikace je možno konvertovat pouze do T-SQL, jazyka typického pro Microsoft SQL Server. Je však možné snadno dodefinovat další druh převodu pomocí rozhraní IConvertable a zajistit tak převod např. do jazyka PL/SQL.

Samotná algoritmizace by měla probíhat dle následujícího diagramu, ovšem může být nutné posloupnosti příkazů mírně pozměnit, tak jak to vyžaduje daný databázový stroj.

[4] (http://msdn.microsoft.com/en-us/library/ms733742.aspx 11.3.2011)
[5] (http://www.codeproject.com/KB/WCF/WCFWebService.aspx 11.3.2011)
[6] (http://www.ingebrigtsen.info/post/2008/11/29/Serialization-in-Silverlight.aspx 11.3.2011)



Obrázek 18: Algoritmizace převodu mezi E-R diagramem a SQL

4.5 Nasazení aplikace

4.5.1 Minimální hardwarová konfigurace

Aplikace není nikterak náročná, ovšem je potřeba počítat s tím, že je síťová a nároky na ní jsou v podstatě dány počtem uživatelů, kteří ji budou naráz používat. Systém je spíš limitován operačním systémem, což je v dnešní době v ideálním případě Windows 2008 R2 edice. Pro minimální běh aplikace je tedy potřeba následující hardwarová konfigurace. [7]

	Minimální konfigurace
CPU	1,4 GHz při jednom jádře nebo 1,3 GHz při dvoujádrovém procesoru
Operační paměť	512 MB
Pevný disk	32 GB pro operační systém, 1GB pro databázi a 50 MB pro aplikaci
Další	Síťová karta (100 Mb/s)

4.5.2 Minimální softwarová konfigurace

Pro běh aplikace je nutno mít i softwarové vybavení. Viz následující tabulka.

Typ software	Název software a verze
Operační systém	Windows XP Professional, Windows Vista Profesional a vyšší, Windows 7 Professional a vyšší, Windows 2003 Server, Windows 2008 Server[8]
Aplikační server	Internet information services 7.5 s podporou Silverlight (nutno doinstalovat) a ASP.NET 4.0 ISAPI filtry
Databázový server	Microsoft SQL Server 2008 R2 v. 10.50.1600.1

4.5.3 Konfigurace aplikace

[7] http://www.microsoft.com/windowsserver2008/en/us/system-requirements.aspx 12.3.2011

5 Popis prostředí aplikace

Aby se tento projekt stal plnohodnotnou webovou aplikací, musí obsahovat jak administrační, tak uživatelské rozhraní, které osobám přistupujícím k aplikaci umožnilo jednoduše využívat veškerou funkčnost kterou nabízí.

5.1 Administrátorské rozhraní - správa uživatelů

Administrátor musí být schopen omezovat přístup k aplikaci. Pouze administrátor může ovlivňovat role uživatelů v systému, neboli může přidělit roli Lector a roli Admin. Změny může administrátor systému provádět v sekci „Správa uživatelů/Upravit uživatele". Pouze administrátor smí mazat uživatele.

Uživatelé s právy Admin a Lector mohou také uživatele přidávat. Přidání uživatele je možno v sekci „Správa uživatelů/Přidat uživatele". Vzhledem k tomu, že nejčastěji se však uživatelé přidávají po určitých kvantech, vždy na začátku kurzu a toto kvantum je generováno z nějakého seznamu studentů, je v aplikaci umožněno přidávat uživatele dávkovým souborem.

Toto přidání je definováno pro seznam studentů vygenerovaný ze systému STAG, který využívá mnoho vysokých škol. Není však problém vytvořit jej i z jiného systému, protože tento soubor musí být ve formátu CSV, který podporuje několik tabulkových procesorů (Microsoft Excel, OpenOffice Calc...). Formát souboru je pevně stanoven ve tvaru Číslo; Příjmení; Jméno; Login. Soubor musí obsahovat tuto hlavičku. Na každé řádce může být pouze jeden student. Jako příklad viz následující část smyšleného souboru.

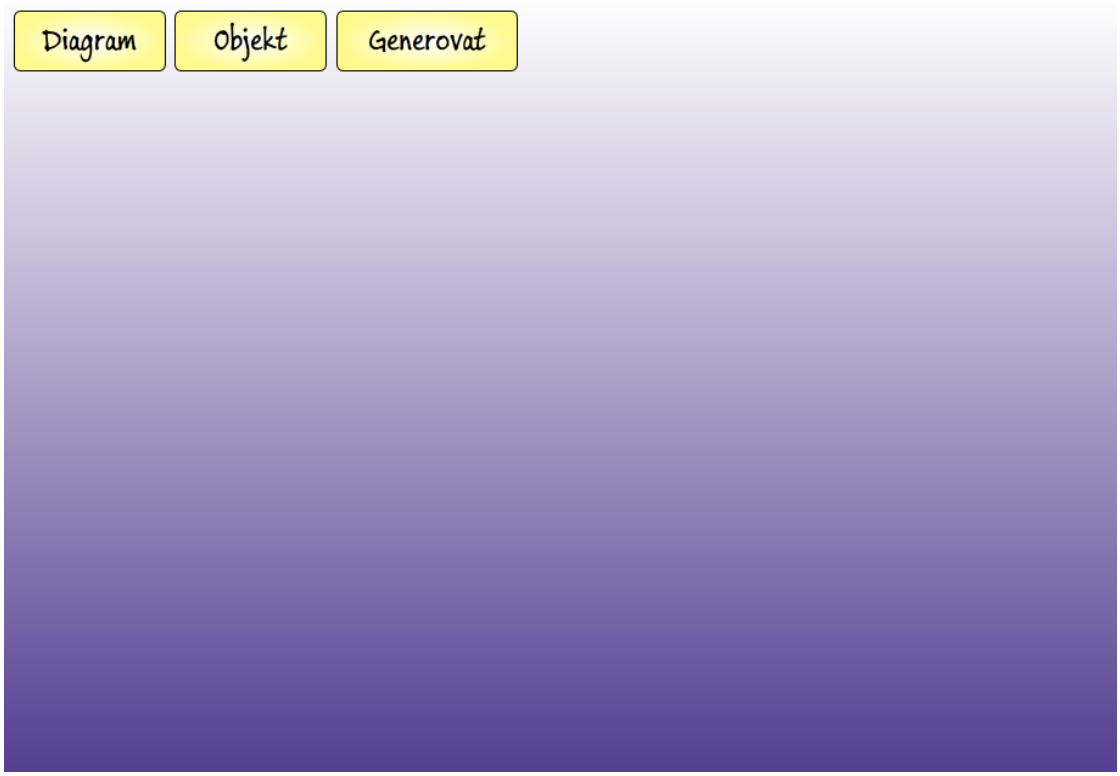
```
Číslo;Příjmení;Jméno;Login
TS07126;Adámek;Bohuslav;adameb00
TS07777;Anděl;Martin;andelm00
TS08222;Balabán;Pavel;balabp00
TS08232;Bergr;Jiří;bergrj00
TS08236;Čajan;Pavel;cajanp00
```

Soubor musí být uložen v kódování UTF-8, kvůli správné reprezentaci české znakové sady. Po vybrání dávkového souboru se asynchronně soubor načte a zobrazí, pro kontrolu vkládaných dat, aby zbytečně nedocházelo k chybám ze strany uživatele.

5.2 Uživatelské rozhraní - Prostředí pro tvorbu diagramů

Prostředí pro tvorbu diagramů je klíčovou částí aplikace, webové rozhraní jen zajišťuje zobrazitelné prostředí pro tuto aplikaci. Ta se nachází v sekci „Editor ERA modelů".

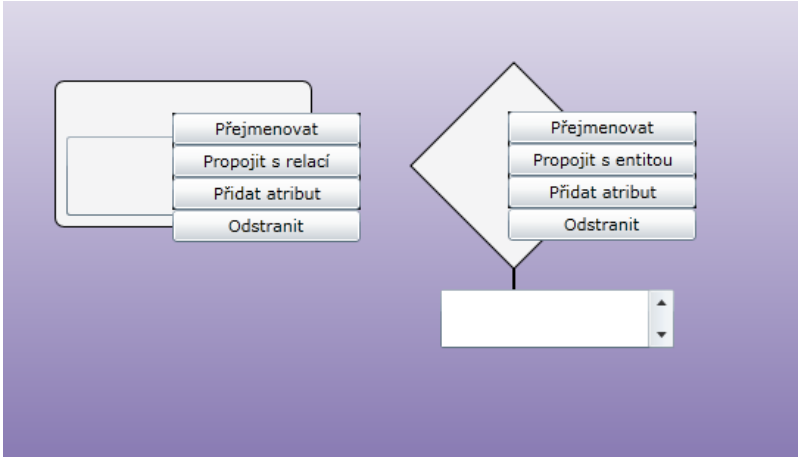
Toto prostředí je vyobrazeno na následujícím obrázku. Postupně rozeberu jeho součásti a ovládání.



Obrázek 19: Aplikace „Editor ERA modelů"

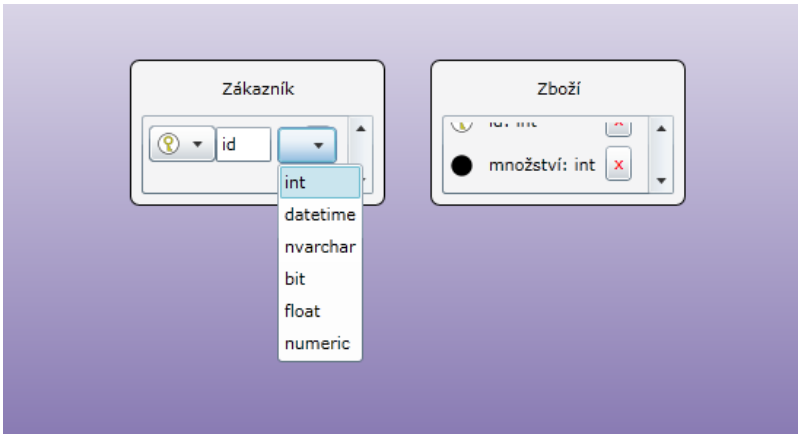
Aplikace obsahuje v horní části menu, které obsahuje možnosti pro ukládání a načítání diagramů, tvorbu objektů entitně relačního diagramu a generování kódu SQL.

Pro definování diagramu jsou v menu „objekt" položky „Vytvořit Entitu" a „Vytvořit Relationship". Pomocí těchto položek se v prostředí objeví buď entita, nebo relace. Tu je pak potřeba blíže specifikovat. Upřesnění informací o daném objektu je přístupné pomocí kliknutí na pravé tlačítko myši na objektu. Objeví se menu, specifické pro daný objekt. Viz následující obrázek.



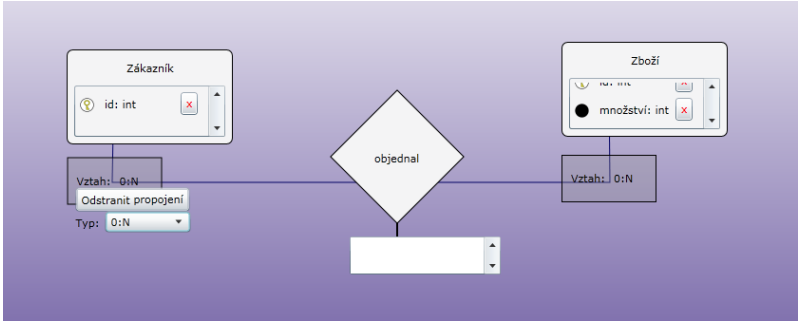
Obrázek 20: Kontextové menu objektů E-R diagramu

Z tohoto kontextového menu je pak možno změnit jméno objektu, propojit jej s jiným objektem, přidat atribut nebo tento objekt odstranit. Pro správné definování entity je potřeba přidat jí primární klíč. Specifikace atributu probíhá v okně uvnitř objektu. Každý atribut má svou specifickou vlastnost (povinný, nepovinný, unikátní, primární klíč), svůj název a datový typ. Atribut je možno kdykoli editovat kliknutím na něj, nebo jej smazat po kliknutí na ikonku s křížkem.



Obrázek 21: Atributy entity

Další nedílnou součástí entitně-relačního diagramu je propojení entity s relací. U tohoto propojení je potřeba definovat kardinalitu vztahu. V kontextovém menu je možné určit kardinalitu, nebo propojení odstranit. Kontextové menu je možno opět vyvolat pravým kliknutím nad čtvercem propojení objektů.



Obrázek 22: Propojení objektů

Po vytvoření diagramu je možné nechat si vygenerovat zdrojový kód SQL, tento kód je možné přímo spustit na databázovém serveru MS-SQL a nechat tento script vytvořit databázi.

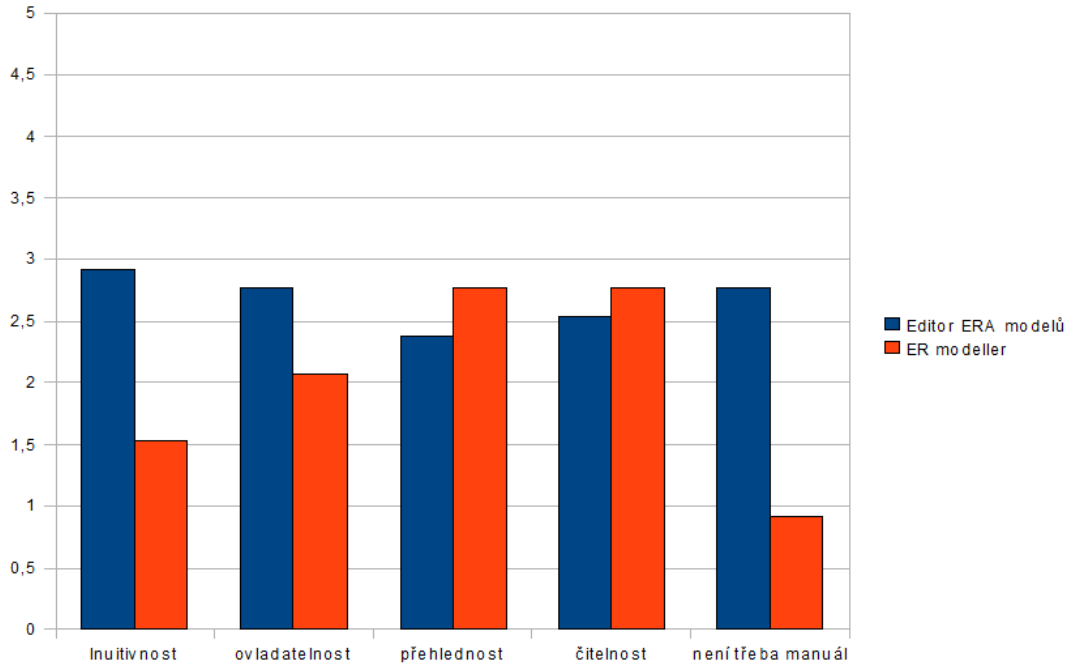
6 Výzkum úspěšnosti vypracování

6.1 Metodika výzkumu

Výzkum jsem prováděl proto, abych objektivně ověřil, zda zpracování aplikace bylo úspěšné. Sběr dat proběhl za pomoci dotazníku. Nejprv respondent vypracoval v tomto editoru diagram podle zadání a pak se jej pokusil realizovat pomocí ER modelleru. Vznikla tak porovnání obou prostředí. Respondenti hodnotili na škále od jedné do pěti. Data jsem pak upravil, aby byl výsledek z grafu zřejmější, pomocí komplementu od čísla 5. Předpokládal jsem, že mé prostředí bude lepší nebo srovnatelné jako ER modeller. Podařilo se mi nasbírat 13 vyplněných dotazníků. Bohužel se mi nepodařilo sehnat více respondentů, protože každý, kdo vyplňuje dotazník, musí mít znalosti entitně-relačního návrhu databáze.

6.2 Výsledky

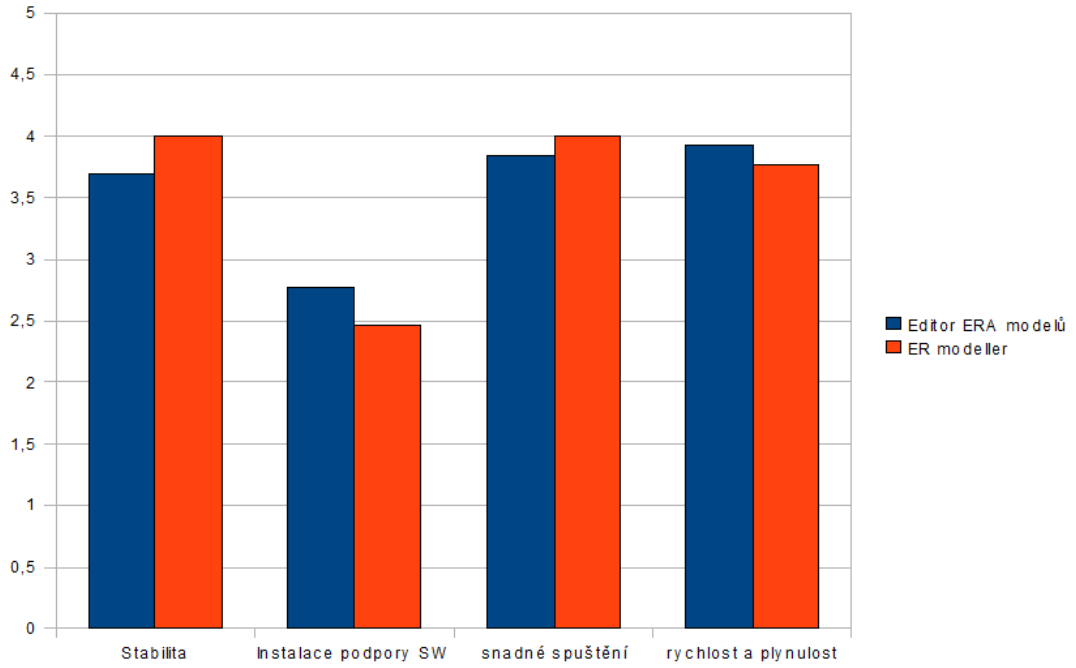
6.2.1 Uživatelská přívětivost



Obrázek 23: Graf - Uživatelská přívětlivost

Z tohoto grafu vyplývá, že mnou vytvořené prostředí je pro uživatele intuitivnější a snáze ovladatelné. Přehlednost a čitelnost dopadly o něco hůře. Z připomínek jsem zjistil, že by uživatelé ocenili, aby okna byla roztahovací a plocha pro graf větší.

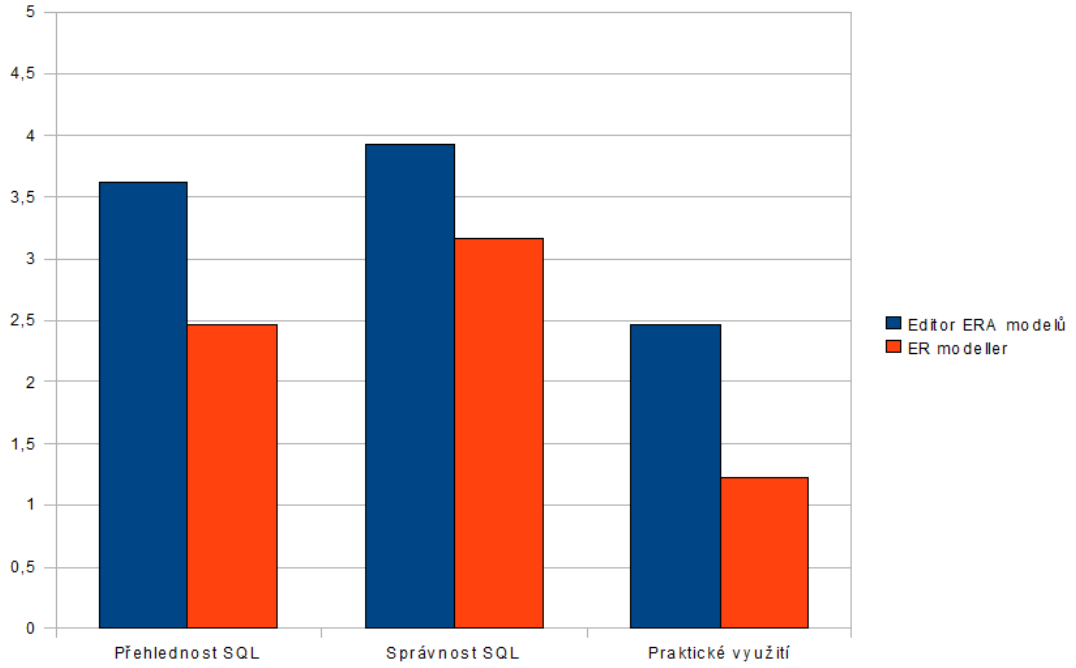
6.2.2 Kompatibilita



Obrázek 24: Graf - Kompatibilita

Kompatibilita dopadla ve všech bodech velmi podobně. Žádné z prostředí nemělo problémy se spuštěním, pády ani rychlostí běhu prostředí. Co mě však překvapilo byl fakt, že více respondentů si museli doinstalovat javu než Silverlight. Domníval jsem se, že právě potřeba instalace Silverlightu bude u mého prostředí problém, ukázalo se však, že tato technologie je již docela rozšířená. Výzkum však může být u této problematiky zkreslen nízkým počtem respondentů a jejich úzké odborné zaměření.

6.2.3 Generování SQL



Obrázek 25: Graf - Generování SQL

Generování SQL dopadlo u mého prostředí lépe. Zdrojový kód je přehlednější a uživatelé se v něm snáze vyznají. Se správností SQL je možno polemizovat, protože respondenti kontrolovali správnost jen od oka, většina z nich nejspíš nezkoušela kód přímo spustit na Oracle serveru, pro který je ER modeller přizpůsoben. Prakticky by mé webové prostředí více tazatelů využilo pro praktické navrhování databází. Je však možné, že někteří z respondentů vůbec entitně-relační model nechce používat.

7 Závěr

Po náročné, ale velice zajímavé práci vzniklo nové prostředí pro tvorbu entitně-relačních diagramů. Všechny cíle se podařilo splnit, prostředí je síťovou aplikací bez potřeby instalace, je schopné spravovat uživatelské účty, diagramy ukládá do databáze a je schopné generovat kód SQL,přizpůsobený pro MS-SQL Server 2008. Po průzkumu úspěšnosti řešení bylo zjištěno, že toto prostředí má oproti aplikaci ER modeller některé výhody, či je s ní srovnatelné.

Při průzkumu jsem zaznamenal několik připomínek ze stran uživatelů, kterým především chyběla možnost zrušení tahu čáry propojení a roztahovací entity. První připomínku jsem do aplikace ihned doplnil. Druhou budu chystat do další verze.

Prostředí budu nadále rozvíjet, sám databáze učím a vím, že takové prostředí je pro studenty potřebné a své uplatnění si jistě najde.

8 Literatura

- [1] FOWLER, Martin. Destilované UML. Praha : Grada, 2009. 176 s. ISBN 978-80-247-2062-3
- [2] LACKO, Luboslav. SQL Hotová řešení. Brno : Computer Press, 2003. 298 s. ISBN 80-7226-975-5
- [3] CONOLLY, Thomas; BEGG, Carolyn; HOLOWCZAK, Richard. Mistrovství - databáze. Brno : Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7
- [4] MACDONALD, Matthew; SZPUSZTA, Mario. ASP.NET 2.0 a C# - tvorba dynamických stránek PROFESIONÁLNĚ. Brno : Zoner press, 2006. 1376 s. ISBN 80-86815-38-2
- [5] ČADA, Ondřej. Objektové programování : naučte se pravidla objektového myšlení. Praha : Grada, 2009. 200 s. ISBN 978-80-247-2745-5

Internetové zdroje

- [6] http://ingebrigtsen.info/post/2008/11/29/Serialization-in-Silverlight.aspx 12.3.2011
- [7] http://www.microsoft.com/windowsserver2008/en/us/system-requirements.aspx 12.3.2011
- [8] http://msdn.microsoft.com/en-us/library/ms733742.aspx 11.3.2011
- [9] http://www.codeproject.com/KB/WCF/WCFWebService.aspx 11.3.2011