UNIVERZITA PARDUBICE

FAKULTA EKONOMICKO-SPRÁVNÍ

BAKALÁŘSKÁ PRÁCE

2021 Markéta Braunová

Univerzita Pardubice
Fakulta ekonomicko-správní

Využití relační databáze pro návrh aplikace

Bakalářská práce

2021 Markéta Braunová

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Markéta Braunová**
Osobní číslo:	**E18360**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Téma práce:	**Využití relační databáze pro návrh aplikace**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování

Cílem práce je vytvoření vzorového příkladu pro předmět Databázové systémy. Vzorový příklad bude vycházet z teoretického pojetí návrhu informačního systému, bude založen na principech relačních databázových systémů a bude využívat nástroje datového a funkčního modelování.

Osnova:
- Základní pojmy související se zpracovávanou problematikou.
- Tvorba modelů.
- Návrh příkladu.

Rozsah pracovní zprávy: **cca 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

KALUŽA, Jindřich a Ludmila KALUŽOVÁ. *Modelování dat v informačních systémech*. Praha: Ekopress, 2012. ISBN 978-80-86929-81-1.
KROENKE, David M. a David J. AUER. *Databáze*. Brno: Computer Press, 2015. ISBN 978-80-251-4352-0.
KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. Brno: Computer Press, 2012. ISBN 978-80-251-1083-6.
POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy*. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

Vedoucí bakalářské práce: **doc. Ing. Stanislava Šimonová, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **1. září 2020**
Termín odevzdání bakalářské práce: **30. dubna 2021**

L.S.

prof. Ing. Jan Stejskal, Ph.D.
děkan

RNDr. Ing. Oldřich Horák, Ph.D.
vedoucí ústavu

V Pardubicích dne 1. září 2020

Prohlašuji:

Práci s názvem Využití relační databáze pro návrh aplikace jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 20. 4. 2021

Markéta Braunová v.r.

PODĚKOVÁNÍ

Mé poděkování patří vedoucí bakalářské práce, doc. Ing. Stanislavě Šimonové, Ph.D., za odborné vedení, za pomoc a rady a za odborný dohled při zpracování této práce. Dále děkuji mé rodině, která mě při tvorbě práce velmi podporovala.

ANOTACE

Cílem této bakalářské práce je vytvoření vzorového příkladu pro předmět Databázové systémy. Vzorový příklad bude vycházet z teoretického pojetí návrhu informačního systému, bude založen na principech relačních databázových systémů a bude využívat nástroje datového a funkčního modelování.

KLÍČOVÁ SLOVA

Databáze, entita, relace, normalizace, relační model, data, návrh databáze, Smart City

TITLE

The use of relation database for application design

ANNOTATION

This bachelor thesis aims to create a sample example for the subject Database Systém. The model example will be built on the theoretical concept of information systém design and based on the principles of relation database systém and will use data and functional modeling tools.

KEYWORDS

Database, entity, relations, normalization, relation model, data, database design, Smart City

OBSAH

ÚVOD

Studium předmětu Databázové systémy, se kterým jsem se měla možnost seznámit ve druhém ročníku studia, mě zaujalo na tolik, že jsem část svého volného času věnovala právě samostudiu databází. Informace pro samotnou tvorbu databází jsem čerpala z mnoha zdrojů a databáze jsem částečně tvořila i v rámci mojí praxe. Díky získaným znalostem, jsem se odvážila tvorbu databází zvolit jako téma pro svou bakalářkou práci. Ráda bych, aby tato práce dokázala usnadnit studium a sloužila jako vzorový příklad pro studenty, kteří se zajímají o tvorbu databází. Práce je strukturována tak, aby se dala použít jako postup při tvorbě databází. V první části vysvětluje teoretické pojmy, které je nezbytné při tvorbě znát. Další část seznamuje studenty s přístupy datového modelování, které je možné použít. Pro tyto dva vybrané přístupy jsou popsány detailní postupy, jak vytvořit finální datový model. Pro snadnější pochopení postupu, jsem si vybrala konkrétní příklad využití databáze. Pro výběr příkladu z reálného života jsem se rozhodla aplikovat odvětví modernizace a digitalizace. Mě osobně, téma modernizace a celkově myšlenky zlepšování kvality a pohodlí života, velmi zajímá. S nadšením využívám různé Smart technologie, díky kterým jsou každodenní aktivity jednodušší a zajímavější. Vidím v nich velké výhody a myslím si, že se stávají součástí našeho každodenního života, ať už v podobě chytrých mobilních telefonů či domácností.

Kromě již zmíněných malých, ale chytrých zařízení, mě také velmi zaujal koncept Smart City. Město Kolín, místo mého bydliště, tuto technologii využívá a přináší tak efektivní řešení pro mnoho každodenních problémů, kterým občané mohou čelit. Konkrétním příkladem, který v mé bakalářské práci podrobněji popisuji, je chytré parkování. Tuto funkci může v dnešní době ocenit většina řidičů, mezi které sama patřím.

Tato práce se věnuje vysvětlení pojmů spojených s tvorbou databází a detailní popsání kroků pro její vytvoření. Tento popis by měl přiblížit co databáze obecně jsou a jak fungují i někomu, kdo se s tímto termínem nikdy předtím nesetkal.

1 VYMEZENÍ TÉMATU PRÁCE

Pro téma bakalářské práce jsem si vybrala ukázku navržení databáze, která má sloužit jako příklad pro studenty předmětu Databázové systémy. V této práci popisuji tvorbu databáze a jaké přístupy datového modelování je možné při tvorbě využít. Práce může také sloužit jako studijní podklad pro studenty, kteří mají o tvorbu databází zájem. Pro příklad jsem se rozhodla využít reálnou problematiku, se kterou se mohou potýkat města a jejich vedení, a to usnadnění současného problému s parkováním využitím konceptu Smart City. Příkladem je tedy ukázka navržení databáze pro chytré parkování, která dokáže zajistit přehlednost o obsazenosti jednotlivých parkovišť ve městě, a také možnost zakoupení parkovacího lístku přes aplikaci. Tato práce z počátku seznámí studenty s pojmem databáze a datové modelování, dále představí dva přístupy datového modelování a konkrétně provede studenty jednotlivými úrovněmi, které vedou k vytvoření finální databáze. Dále se tato práce věnuje představení konceptu Smart City, které bude následně využito pro konkrétní příklad. Poslední částí práce představuje ukázku celého procesu tvorby databáze chytrého parkování. Tento proces zahrnuje určení vstupních dat, která poskytuje zájemce o databázi a dále již samotná tvorba databáze v jednotlivých úrovní. Databáze chytrého parkování bude navržena pomocí kombinace strukturovaného a objektově-orientovaného přístupu datového modelování.

Téma navržení databáze jako učební příklad jsem si vybrala ve snaze popsat jednotlivé kroky tvorby databáze pro studenty, kteří jsou začátečníky v tomto oboru. Touto prací bych ráda usnadnila pochopení tvorby jednotlivých databází, a zároveň ukázala, že databáze je možné využít i při řešení modernizace města pomocí konceptu Smart City.

1.1 DATABÁZOVÉ SYSTÉMY

Pro vytvoření aplikace chytrého parkování je nejprve zapotřebí provést návrh relační databáze chytrého parkování.

Obecně databáze představuje místo, kam se ukládají data. Zjednodušeně si ji lze představit jako kartotéku například u lékaře, ve které jsou zařazeny složky s daty, která představují tzv. entity a ve složkách jsou určité informace (například jméno a příjmení pacienta, adresa apod.), které se nazývají atributy. (Pokorný, 2020)

Relační databáze je pak určitý typ databáze, která propojuje datové atributy pomocí jedinečného ID, nazvaného klíč. Tyto databáze se dnes již hojně využívají ve všech odvětví,

kde je důležité zachovat bezpečnost a konzistenci dat pomocí definovaných pravidel. Tento typ databáze podporuje logické operace nad daty tzv. transakce, které musí splňovat čtyři základní vlastnosti: (Co je relační databáze?, 2021)

- Atomicita – říká, že se provedou všechny operace s entitami najednou, anebo žádná a entity zůstanou v původním stavu. Pokud tedy z nějakého důvodu neproběhne celá transakce (například z důvodu výpadku proudu apod.) je garantováno, že neproběhnou žádné změny, tedy vše se vrátí do původního stavu před prováděnou transakcí.
- Konzistence – převede databázi z jednoho konzistentního stavu do druhého bez porušení integritních omezení. Zamezuje například pouze jednostrannému převodu peněz, kdy musí dojít jednak k odečtení peněz z jednoho účtu a zároveň přičtení peněz k účtu druhému. (Kopal, 2015)
- Izolace – zaručuje, že operace s daty budou skryty před ostatními operacemi, dokud nebude celá transakce úspěšně provedena.
- Trvalost – zajistí, že se provedené změny dat trvale uloží v databázi po úspěšném provedení transakce. (Co je relační databáze?, 2021)

1.2 DATOVÉ MODELOVÁNÍ

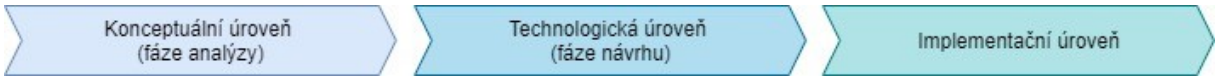
Pro vytvoření aplikace chytrého parkování je zapotřebí navrhnout databázový systém, který bude daná aplikace využívat pro ukládání dat (Zelenka, 2004). Principem datového modelování je nalézt objekty (tzv. entity) požadovaného systému a určit mezi nimi vztahy. U nalezených objektů je také důležité určit jejich vlastnosti (tzv. atributy). Po sesbírání těchto informací pomocí datového modelování vytváříme datový model, který slouží jako podklad pro vytvoření výsledné databáze. (Skřivan, 2008)

Pro popis procesu vývoje datového modelování je potřeba využít tří úrovňový princip architektur, který umožňuje reagovat na případné změny v databázi. Jedná se o konceptuální, technologickou a implementační úroveň.

V **konceptuální úrovni**, jinak zvané také fázi analýzy, dochází k analýze požadavků na danou databázi. Po určení požadavků se vytváří počáteční datový model, ve kterém jsou definovaná data, jejich vztahy a omezení pro splnění příslušných požadavků na databázový systém.

Další úrovní je **technologická úroveň**, též nazvaná jako fáze návrhu. Tato úroveň umožňuje vytvořit datový model, který bude připraven k implementaci do daného nástroje pro vytvoření finální databáze. Obecně tato úroveň přiřazuje k datovému modelu tzv. klíče, které zajistí propojenost mezi jednotlivými tabulkami. V této fázi dochází k transformaci datového modelu a následně k jeho normalizaci, tedy k odstranění anomálií v databázi.

Poslední fází vývoje databáze je **implementační úroveň**, ve které dochází k implementaci dané databáze, tedy zavedení hotové databáze do provozu. (Kroenke, 2015)



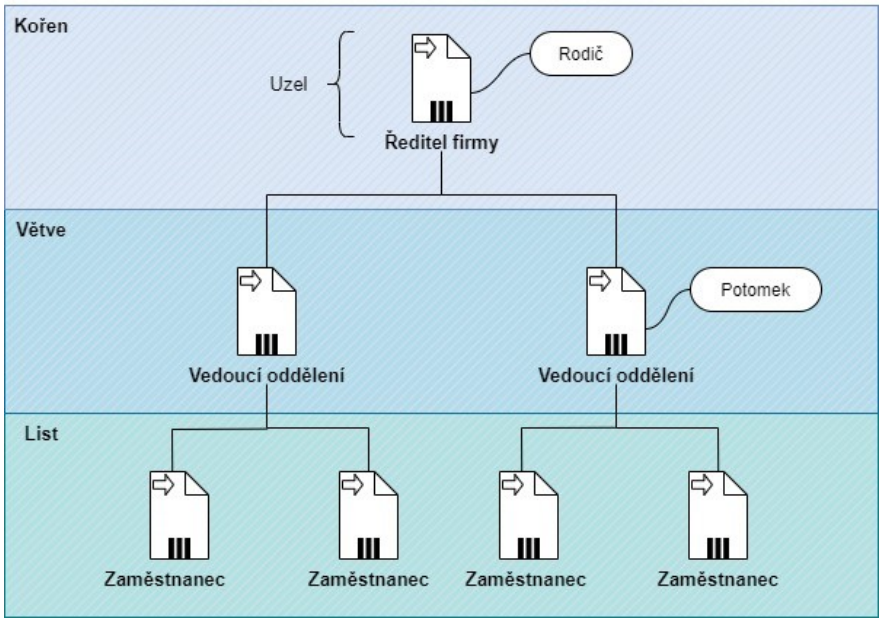
Obrázek 1: Úrovně architektury

Zdroj: [vlastní zpracování]

Během vývoje byly vyvinuty 4 typy datových modelů, které je možné kombinovat (například relační a objektový). Zvolení datového modelu závisí na tom, ve kterém se nejlépe bude daná databáze vytvářet. Nejpoužívanějšími modely v současnosti jsou relační a objektové modely. (Kaluža, 2012)

Hierarchický datový model je nejstarším modelem pro tvorbu databáze a jeho první verze byla spuštěna v roce 1968. Podstatou tohoto modelu je stromová struktura modelovaných dat, tedy data jsou uspořádána do převrácené stromové struktury, která je hierarchicky seřazená (Kaluža, 2012). Záznamy o konkrétním objektu se v tomto modelu nazývají uzly. Mezi jednotlivými uzly jsou určeny vztahy, které jsou pak definovány termíny rodič a potomek. Rodičovský uzel se může seskupovat s jedním nebo více uzly označovanými potomek, zato potomek může být sdružován pouze s jedním rodičovským uzlem. Uzly se v modelu řadí do různých úrovní, podle toho, v jaké úrovni stromové struktury se nacházejí:

- Uzel, který je umístěn nejvýše v hierarchii a nemá nad sebou žádného rodiče se nazývá **kořen**.
- Další úrovní ve stromové struktuře je **větev**, kam spadá uzel, který má nad sebou rodičovský uzel a pod sebou jednoho nebo více uzlů potomků.
- Poslední úrovní je **list**, do které patří uzly, které mají nad sebou rodičovský uzel a pod sebou nemají žádné potomky. (Kybkalo, 2013)



Obrázek 2: Příklad hierarchického datového modelu

Zdroj: [vlastní zpracování]

Největší nevýhodou hierarchické modelu je složité vkládání a odstraňování záznamů. V případě, že by došlo k odstranění rodičovského uzlu, dojde také k odstranění všech jeho potomků.

Síťový datový model zdokonaluje hierarchický model a zobecňuje jeho koncepci. Výhodou tohoto modelu oproti předchozímu je rychlejší přístup k datům a doplnění o mnohonásobné vztahy nazývané sety. Sety provádějí propojení záznamů různého nebo stejného typu a lze je spojit s jedním nebo více záznamy. Nevýhodou síťového modelu je obtížná změna jeho struktury, v některých případech je i potřeba vytvořit celý model od začátku (Databázové modely, 2010). Síťový datový model lze využít například pro geografické informační systémy, ve kterých se zpracovávají geografické údaje. (Skřivan, 2008)

Relační datový model je v současnosti nejpoužívanějším datovým modelem. Tento model má jednoduchou strukturu, která ukládá data do jednotlivých tabulek. Tyto tabulky mohou být na sebe závislé, tedy dochází zde k propojení konkrétních datových polí v různých tabulkách. (Databázové modely, 2010)

Objektový datový model vznikl z důvodu, aby umožnil ukládat a dále zpracovávat objekty podle specifických potřeb aplikací, což relační datový model neumožňoval (Kybkalo, 2013). Tento datový model lze využít například pro aplikace, které jsou vytvářené

pomocí objektově orientovaného programování, což pro předchozí model není vhodné. Nevýhodou tohoto modelu je jeho složitost, proto se ve většině případech objektový model využívá v kombinaci s relačním datovým modelem. (Skřivan, 2008)

1.3 VÝVOJ RELAČNÍHO DATOVÉHO MODELU

Pro vývoj relačního modelu je důležité zvolit přístup modelování, ve kterém bude daná databáze vytvářena a vznikne tak výsledný relační model. Každý přístup nahlíží na systém odlišně a využívá rozdílné diagramy.

Strukturovaný přístup modelování se využívá od 70. let 20. století a přistupuje k systému zvlášť z pohledu dat a zvlášť z pohledu funkcí. V tomto přístupu se využívá v konceptuální úrovni ER diagram a v úrovni technologické se tento diagram transformuje do výsledného relačního datového modelu. (Danel, 2013)

Objektově – orientovaný přístup modelování je novější přístup, který se využívá od 90. let 20. století. Oproti strukturovanému přístupu tento přístup zachycuje data a funkce v jednom objektu. Objektově – orientovaný přístup je založen na objektech, které mají určité vlastnosti a operace, které daný objekt může vykonávat. V tomto přístupu lze využít Use Case diagram, Class diagram nebo například Sekvenční diagram. (Matula, 2015)

V další kapitole jsou oba přístupy podrobně popsány a rozebrány v jednotlivých úrovní, tedy v konceptuální a technologické úrovni.

1.4 STRUKTUROVANÝ PŘÍSTUP MODELOVÁNÍ

1.4.1 KONCEPTUÁLNÍ ÚROVEŇ

Hlavním cílem konceptuální úrovně je dostatečně vystihnout popis reality a o řešení konkrétního systému se stará následující úroveň. Tato úroveň představuje sepsání požadavků na systém a jakýsi prototyp modelu, který je v dalších fázích rozvíjen a implementován do konečné podoby. Ve fázi analýzy se vytváří datový model pomocí ER modelu, což v překladu znamená model entit a vztahů. Prvky, které tvoří ER model jsou entity a vztahy mezi nimi, atributy, primární klíč, datové typy a integritní omezení. V této části jsou představeny jednotlivé prvky a v kapitole 7.3 je ukázán konkrétní příklad ER modelu.

ENTITY

Základem každé relační databáze je tabulka, ve které jsou uložena potřebná data. Tato tabulka je nazývána entita. Entita představuje něco, co uživatelé mají v plánu sledovat. Je to cokoliv, o čem chce uživatel v sytému zaznamenávat informace. (Kroenke, 2015)

Entitou mohou být jednak hmotné objekty, jako je například *Zákazník*, *Dodavatel* nebo *Zaměstnanec*, ale také nehmotné objekty, kdy příkladem může být *Objednávka*, *Výpůjčka* nebo *Adresa*. (Skřivan, 2008)

ATRIBUTY

Každá entita (tabulka) se skládá z několika řádků, které se nazývají atributy. Atributy jsou nějaké vlastnosti entity, které jí přiřazují hodnotu. Do těchto atributů se pak dále vkládají záznamy, tedy nějaká konkrétní data dané entity nazývané výskyty. (Kroenke, 2015)

Příklad: Pokud se vytvoří entita *Zaměstnanec* jsou jejími atributy například *Jméno zaměstnance*, *Datum narození zaměstnance* nebo například jeho *Adresa*. Výskytem pak například u atributu *Jméno zaměstnance* je *Pavel Novák*.

DATOVÉ TYPY

Jednotlivé atributy mají datové typy, což jsou hodnoty, kterých mohou záznamy v tabulce nabývat. Existuje nespočet datových typů, mezi nejdůležitější však patří datový typ číslo, text, datum a čas a doména.

- **Číslo** – jedná se o číselný datový typ, kde je vhodné uvést omezení, tedy kolik čísel je možné vložit do tohoto datového typu. Příkladem může být atribut *Telefonní číslo*, kdy se počet čísel bez předvolby nastavuje na 9.
- **Text** – představuje textový datový typ, kam je možné vkládat jakékoliv znaky (písmena, symboly i čísla). Opět je vhodné nastavit délku textu, tedy kolik znaků může mít daný záznam. V případě, že záznam má obsahovat text a zároveň i čísla, například atribut *RZ* (registrační značka), lze využít tento datový typ.
- **Datum a čas** – tento datový typ slouží k vepsání datumu a času.
- **Doména** – jedná se o odlišný datový typ, který představuje seznam správných hodnot, kterých může atribut nabývat. V praxi to lze přirovnat k roletkovému menu, ze kterého je vybrána vhodná možnost. Příkladem může být například atribut *Dosažené vzdělání* u entity *Zaměstnanec*, kde u atributu bude vytvořena

doména s možnostmi například *Střední vzdělání s maturitou*, *Vyšší odborné vzdělání* nebo *Vysokoškolské vzdělání*. (Microsoft 365, 2021)

PRIMÁRNÍ KLÍČ

Primární klíč zaručuje jednoznačnou identifikaci záznamů. Zajišťuje, aby jednotlivé záznamy byly od sebe odlišené. Jako primární klíč se označuje jedinečný atribut, v případě, že jeden atribut k jedinečnosti nestačí, lze použít více atributů a označit je jako složený primární klíč. V případě, že žádný z atributů nemůže sloužit jako primární klíč, je možné vytvořit takzvaný umělý primární klíč, který svou jedinečnou hodnotu automaticky vygeneruje. Příkladem umělého primárního klíče může být *Číslo zaměstnance*. (Kaluža, 2012)

Příklad: Je vytvořená entita *Zaměstnanec*, která obsahuje atributy *Jméno*, *Příjmení* a *Rodné číslo*. Problém může nastat, když na pracovišti budou dva zaměstnanci se stejným jménem a příjmením. Je nutné je od sebe nějakým způsobem odlišit, a to je provedeno právě pomocí primárního klíče. Atribut *Rodné číslo* se tak označí jako primární klíč, protože každý člověk má jedinečné rodné číslo.

VZTAHY MEZI ENTITAMI

V databázi se vyskytuje více entit, které mohou být vzájemně propojené, tedy je mezi nimi určitá vazba. Vztahy mezi entitami se určují dvěma způsoby, a to kardinalitou a parcialitou. Parcialita zachycuje, zda všechny výskyty entity musí nebo nemusí být zapojeny do příslušného stavu (povinnost je značena 1 a nepovinnost 0). Kardinalita určuje, kolik daný vztah obsahuje výskytů entity (jeden výskyt je značen 1, více značeno N). (Kaluža, 2012)

Nejpoužívanější typy vazeb mezi entitami jsou ukázány na příkladech.

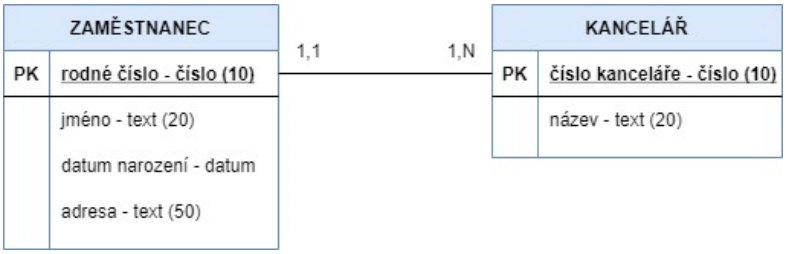
Vazba 1,1 a 1,1

Obrázek 3: Ukázka vazby 1,1 a 1,1 mezi entitami

Zdroj: [vlastní zpracování]

Na příkladě je uvedena entita *Zaměstnanec* a entita *Pracovní stůl*. Každý *Zaměstnanec* pracuje u jednoho stolu, použije se zde tedy vazba 1,1. *Pracovní stůl* je přiřazen pouze jednomu *Zaměstnanci*, proto opět vazba 1,1.

Vazba 1,1 a 1,N

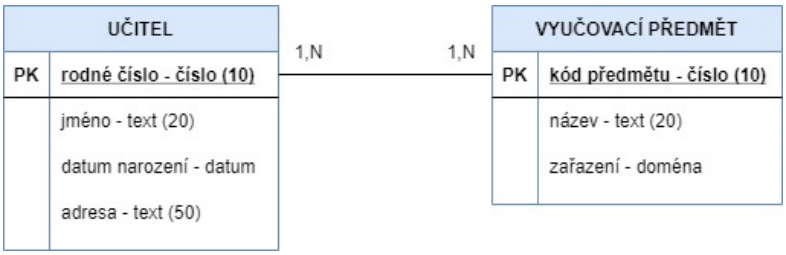


Obrázek 4: Ukázka vazby 1,1 a 1,N mezi entitami

Zdroj: [vlastní zpracování]

Na tomto příkladě je vytvořená entita *Zaměstnanec* a entita *Kancelář*. *Zaměstnanec* musí mít přidělenou právě jednu kancelář, proto je u této entity vazba 1,1. Zato *Kancelář* musí mít alespoň jednoho zaměstnance, ale může jich tam pracovat i více, proto vazba u této entity je 1,N.

Vazba 1,N a 1,N



Obrázek 5: Ukázka vazby 1,N a 1,N mezi entitami

Zdroj: [vlastní zpracování]

Tento příklad zobrazuje entitu *Učitel* a entitu *Vyučovací předmět*. *Učitel* musí vyučovat alespoň jeden předmět, čemuž odpovídá vazba 1,N a *Vyučovací předmět* musí být vyučován alespoň jedním učitelem, proto opět vazba 1,N. (Vávra, 2012)

1.4.2 TECHNOLOGICKÁ ÚROVEŇ

Jak už je výše popsáno v předchozí úrovni se zjistily požadavky na danou databázi a vytvořil se základní ER model. V technologické úrovni se tento model transformuje do relačního datového modelu, který je možný implementovat a vytvořit navazující aplikaci. Ještě před zpřístupněním modelu je důležité provést normalizaci, ve které dochází ke kontrole, zda se v relačním modelu nevyskytují anomálie.

Prvky, které se objeví ve fázi návrhu jsou stejné jako ve fázi analýzy. Jedná se o atributy, domény a primární klíč. Nově v této úrovni vznikají relace a vazby mezi nimi a cizí klíče. V této části opět představím jednotlivé prvky a v kapitole 8 ukážu příklad relačního modelu. (Kaluža, 2012)

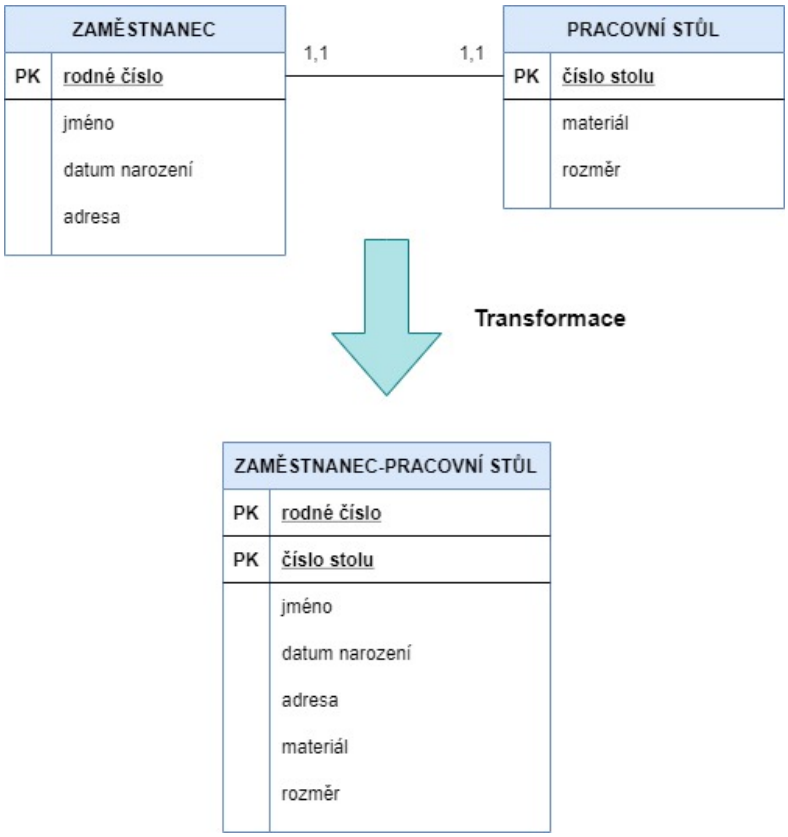
CIZÍ KLÍČ

Cizí klíč představuje hodnotu primárního klíče v entitě, která se při transformaci vytvoří v jiné relaci a umožňuje tak propojení dvou relací pomocí primárního a cizího klíče. Funkce cizího klíče je zobrazena níže při popisu relací a jejich vazeb.

RELACE A VAZBY MEZI NIMI

Relacemi se nazývají tabulky, které vznikají po propojení entit pomocí vazeb. Jak je popsáno výše existují tři nejpoužívanější vazby mezi entitami, pomocí kterých se vytvářejí jednotlivé relace podle stanovených pravidel. (Kroenke, 2015)

Vazba 1,1 a 1,1 mezi entitami vytvoří pouze jednu relaci, do které se přesunou všechny atributy z obou entit. K lepšímu pochopení je zde ukázka vzniku relace, a to na příkladu z konceptuální úrovně.

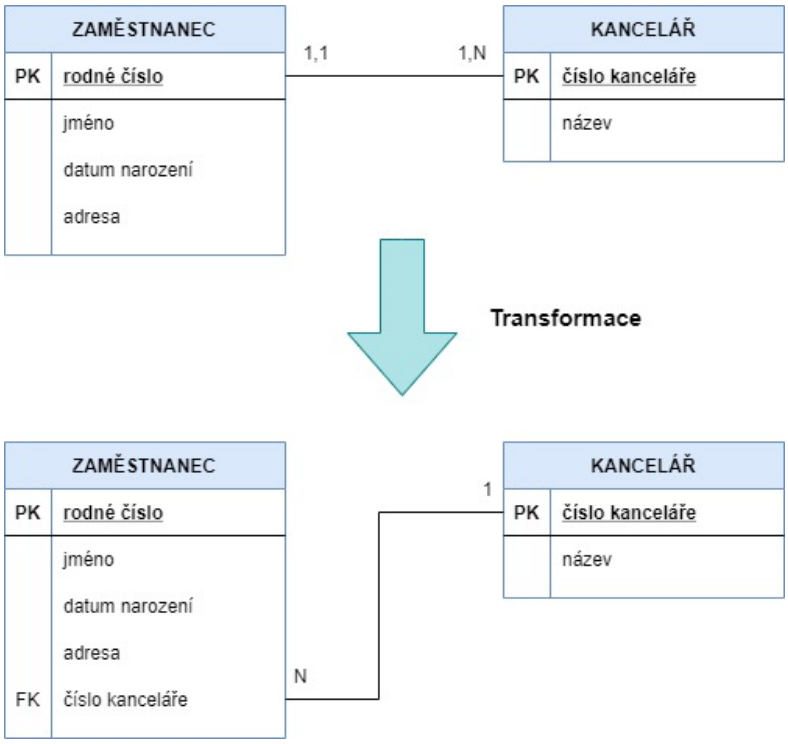


Obrázek 6: Transformace na relace z vazby 1,1 a 1,1

Zdroj: [vlastní zpracování]

U entity *Zaměstnanec* a *Pracovní stůl* je přiřazena vazba 1,1, proto dojde k vytvoření jedné relace *Zaměstnanec-Pracovní stůl*, do které se přesunou veškeré atributy i oba primární klíče.

Vazba 1,1 a 1,N mezi entitami vytvoří dvě relace, které jsou propojené pomocí cizího klíče. Primární klíč v relaci, u které je vazba 1,N se stane cizím klíčem v relaci s vazbou 1,1 a vzájemně se tyto klíče propojí. Opět k lepšímu pochopení je zde ukázka vzniku relací na příkladu z konceptuální úrovně.

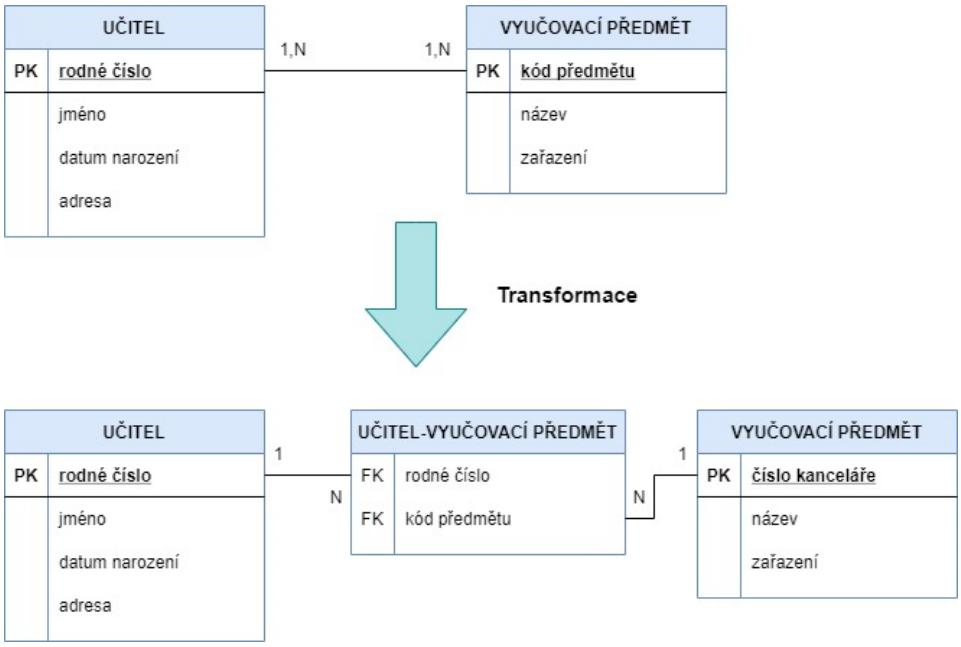


Obrázek 7: Transformace na relace z vazby 1,1 a 1,N

Zdroj: [vlastní zpracování]

U entity *Zaměstnanec* je vazba 1,1 a u entity *Kancelář* je vazba 1,N, proto dojde ke vzniku dvou relací. V relaci *Zaměstnanec*, kde je vazba 1,1 vzniká cizí klíč *Číslo kanceláře* z primárního klíče v relaci *Kancelář*. Pomocí primárního a cizího klíče jsou poté tyto relace propojené.

Vazba 1,N a 1,N mezi entitami vytvoří tři relace, kdy třetí se stává vazební, což znamená, že v této relaci vznikají cizí klíče ze dvou zbylých relací. Pomocí těchto klíčů jsou opět propojené všechny tři relace.



Obrázek 8: Transformace na relace z vazby 1,N a 1,N

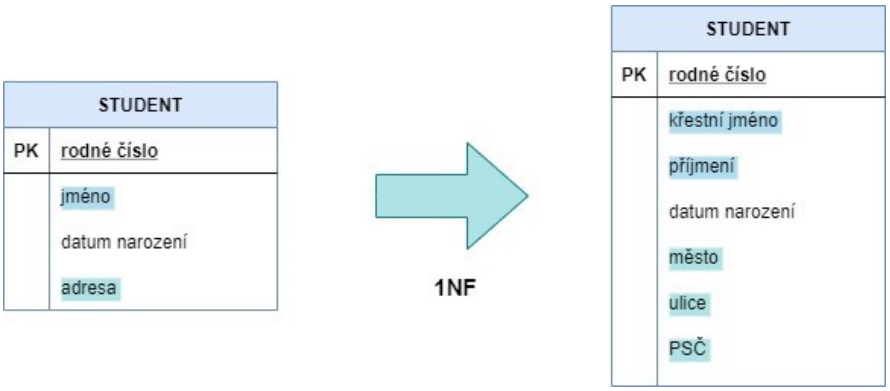
Zdroj: [vlastní zpracování]

U entity *Učitel* a *Vyučovací předmět* je přiřazena vazba 1,N, proto dojde k vytvoření tří relací. Prostřední relace *Učitel-Vyučovací předmět* je vazební relací a vznikají v ní dva cizí klíče *Rodné číslo učitele* a *Kód předmětu*, který jsou primárními klíči v relacích *Učitel* a *Vyučovací předmět*. Pomocí primárních a cizích klíčů jsou následně tyto relace mezi sebou propojeny. (Microsoft 365, 2021)

NORMALIZACE

Vstupem do normalizace je již vytvořený relační datový model, ve kterém se hledají chyby neboli anomálie. Tyto anomálie software neumí realizovat a k jejich odhalení slouží normální formy (značené NF). Tento proces je velmi důležitý pro správné navržení a funkčnost databáze. Výstupem z normalizace bude tedy vyčištěný relační datový model, který je připraven na implementaci. Existuje řada normálních forem, ale v praxi se nejvíce využívají především první tři normální formy.

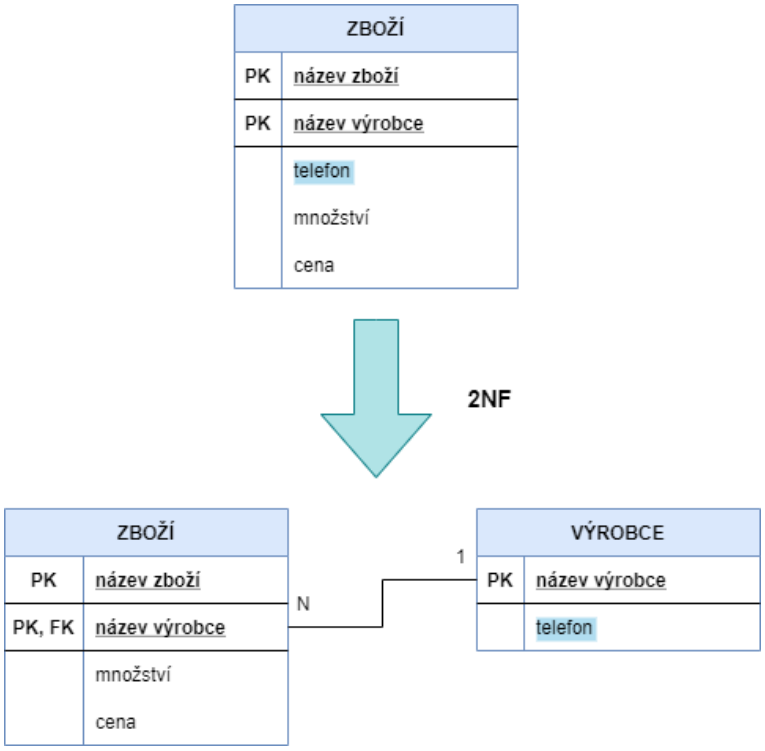
- **1NF** – první normální forma říká, že všechny atributy v relacích jsou dále nedělitelné. Vhodným příkladem pro upřesnění této normální formy je atribut *Jméno*, které lze rozdělit na *Křesní jméno* a *Příjmení*. Pokud by některá relace obsahovala tuto anomálii znamenalo by to, že nesplňuje první normální formu a je nutné daný atribut rozdělit na více atributů.



Obrázek 9: Odhalení anomálie pomocí 1NF

Zdroj: [vlastní zpracování]

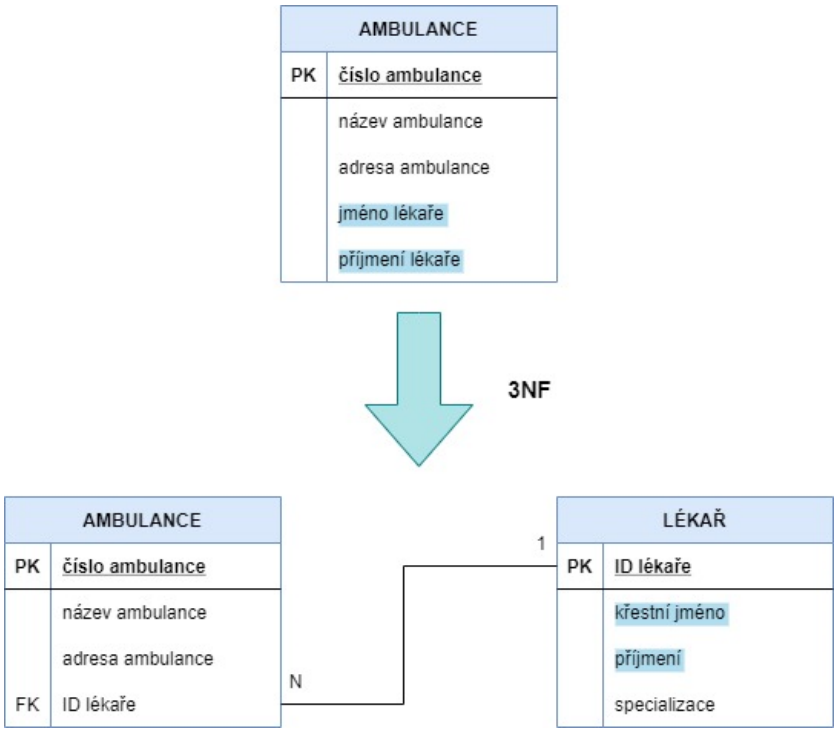
- **2NF** – pro splnění druhé normální formy je nutné nejprve splnit podmínku první normální formy a dále musí být všechny atributy v dané relaci závislé na celém složeném primárním klíči. Nesmí tedy nastat případ, kdy by některý z atributů závisel pouze na části složeného primárního klíče. V případě, že v relaci není složený primární klíč, automaticky je tato podmínka splněna. Pokud se odhalí anomálie pomocí této normální formy je nutné tuto relaci rozdělit do více tabulek.



Obrázek 10: Odhalení anomálie pomocí 2NF

Zdroj: [vlastní zpracování]

- **3NF** – ke splnění třetí normální formy je nutné splnit předchozí dvě normální formy, a navíc musí jednotlivé atributy záviset pouze na primárním klíči, ne na ostatní neklíčových atributech. V případě, že tato podmínka nebude splněna je nutné opět rozdělit relaci do více tabulek. (Vávra, 2012)



Obrázek 11: Odhalení anomálie pomocí 3NF

Zdroj: [vlastní zpracování]

1.5 OBJEKTOVĚ – ORIENTOVANÝ PŘÍSTUP MODELOVÁNÍ

1.5.1 KONCEPTUÁLNÍ ÚROVEŇ

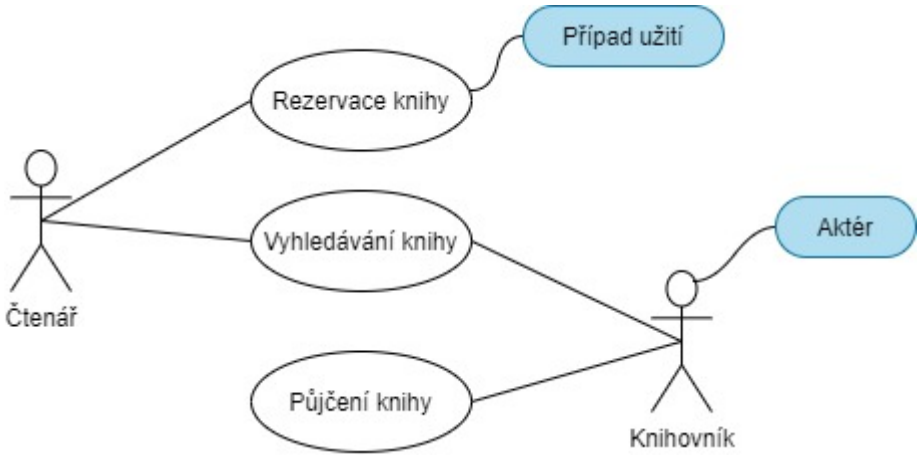
Stejně jako u strukturovaného přístupu modelování, je hlavním cílem konceptuální úrovně správné pochopení uživatelských požadavků na systém. Ve fázi návrhu se tyto požadavky formují do různých diagramů, které jsou popsány v této kapitole. (Fowler, 2019)

USE CASE

Use Case diagram není nutné použít, ale v případě, že daná databáze má sloužit k vytvoření následné aplikace, je tento diagram vhodné navrhnout. Use Case zachycuje funkčnost budoucí aplikace a rozsah oprávnění jednotlivých uživatelů (tzv. aktérů) systému.

Tento diagram umožňuje vývojářům databáze lépe pochopit požadavky uživatelů na systém. Součástí Use Case diagramu jsou aktéři, jednotlivé případy užití a k nim vytvořené scénáře.

Aktér představuje roli uživatele, který komunikuje s vytvořeným systémem. Příkladem aktéra v databázi může být například *Správce aplikace*, *Zaměstnanec* nebo *Uživatel*, ale také aktérem nemusí být člověk, ale například jiný *Systém*. S aktéry se váže termín **Případy užití**, což jsou činnosti, které v systému může vykonávat aktér. V daném systému může jeden aktér vykonávat jeden nebo více případů užití a zároveň jeden případ užití může provozovat jeden nebo více aktérů. Jednotlivé případy užití by měly být stručné, srozumitelné a vyjádřeny slovesnou vazbou, například *Registrace uživatele* nebo *Úprava evidence zákazníků*. Tyto případy užití jsou dále rozvinuty **scénáři**, které podrobně popisují, jak daná činnost bude probíhat v aplikaci. Scénáře vyjadřují postupnost kroků, které popisují chování aktéra a systému v dané aplikaci. Ke hlavnímu úspěšnému scénáři je nutné vždy doplnit také scénář alternativní, který udává případné rozšíření kroků nebo jiná řešení v případě neúspěšného provedení hlavního scénáře. (Kanisová, 2012)



Obrázek 12: Ukázka Use Case diagramu

Zdroj: [vlastní zpracování]

Název Use Case	**Rezervace knihy**
Aktér	Čtenář
Popis	Postup, jak si čtenář může rezervovat knihu
Počáteční stav	Čtenář si chce zarezervovat knihu
Konečný stav	Úspěšná rezervace knihy
Spouštěcí událost	Výběr konkrétní knihy
Hlavní scénář	
Krok 1:	A: Vybere konkrétní knihu a klikne na tlačítko „Rezervovat“
Krok 2:	S: Zobrazí formulář pro rezervaci knihy, kde vyplní pole, která se týkají knihy

Krok 3:	A: Vyplní zbylé informace ve formuláři (své jméno a příjmení, adresu, datum narození atd.) a klikne na tlačítko „Dokončit rezervaci“
Krok 4:	S: Zobrazí shrnutí rezervace
Krok 5:	A: Zkontroluje shrnutí rezervace a klikne na tlačítko „Dokončit“
Krok 6:	S: Odešle vyplněný formulář s rezervací knihy do systému a shrnutí objednávky čtenářovi
Krok 7:	S: Změní stav konkrétní knihy na „Rezervováno“
Alternativní scénář	
Krok 3:	Aktér chybně vyplní údaje do formuláře S: Napíše upozorňující hlášku „Chybně vyplněné údaje“ a zamezí odeslání formuláře A: Opraví chybně vyplněné údaje
Krok 3:	Aktér nevyplní všechny údaje do formuláře S: Napíše upozorňující hlášku „Nevyplněné pole formuláře“ a zamezí odeslání formuláře A: Vyplní prázdné pole ve formuláři

Tabulka 1: Ukázka scénáře k Use Case diagramu

Zdroj: [vlastní zpracování]

CLASS DIAGRAM

Class diagram neboli diagram tříd znázorňuje jednotlivé objekty v systému a vztahy mezi nimi. Dalo by se říci, že Class diagram je podobný ER diagramu ve strukturovaném přístupu modelování. Důvod jeho vývoje je vytvoření co nejvěrnějšího modelování reality. ER diagram je stále velmi využívaný, ale chybí mu přítomnost operací, které vyjadřují chování jednotlivých objektů třídy. Tyto operace právě diagram tříd umožňuje vytvářet. Class diagram se vytváří jednak ve fázi analýzy, tak ve fázi návrhu, kde se rozšiřuje o některé vlastnosti. Součástí tohoto diagramu jsou třídy a vazby mezi nimi, dále atributy, objekty a operace. V konceptuální úrovni se vytváří model, který vzniká propojením vztahů mezi třídami, čehož se docílí pomocí vazeb.

Třída odpovídá ve strukturovaném přístupu entitě, představuje tedy něco, o čem chce uživatel systému uchovávat informace, například třída *Zákazník*. Součástí třídy jsou **atributy**, které mají stejnou funkčnost jako u strukturovaného přístupu modelování, tudíž to jsou vlastnosti dané třídy, které jí přiřazují hodnotu. Příkladem atributu může být *Jméno zákazníka*. Do těchto atributů se následně ukládají jednotlivé záznamy, které představují nějaká konkrétní data dané třídy nazývané **objekty**. Opět tento prvek odpovídá ve strukturovaném přístupu výskytům entity, tedy objektem u třídy *Zákazník* může být *Petr Nový*. (Kaluža, 2012)

V databázi se vyskytuje více tříd, které mohou být mezi sebou vzájemně propojené, stejně jako u relačního datového modelu. Toto propojení se uskutečňuje pomocí **násobnosti**,

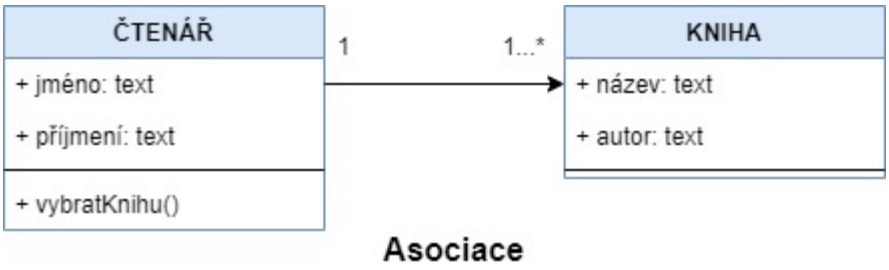
která určuje, kolik objektů se může účastnit příslušného vztahu a vyjadřuje se horní a spodní hranicí.

- Právě jeden (značen 1) – tato násobnost říká, že příslušného vztahu se může účastnit pouze jeden objekt.
- Jeden nebo více (značen 1…*) – toto označení určuje, že daný vztah může obsahovat jeden nebo více objektů.
- Nula nebo více (značen 0…*) – v tomto případě daný vztah může obsahovat nula nebo více objektů. (Fowler, 2019)

Datové typy atributů se shodují s datovými typy ve strukturovaném přístupu modelování. Z atributů, u kterých by byl přiřazen datový typ doména se stane výčtová třída (<<*enumeration*>>), kde se vypíše seznam možných hodnot atributů. (Urban, 2003)

Dále se určují vztahy mezi jednotlivými třídami na úrovni instance, a to asociace, která se v diagramu značí čárou orientovanou ze zdrojové do cílové třídy, dále agregace značená čárou s bílým kosočtvercem a kompozice vyjádřená čárou s černým kosočtvercem.

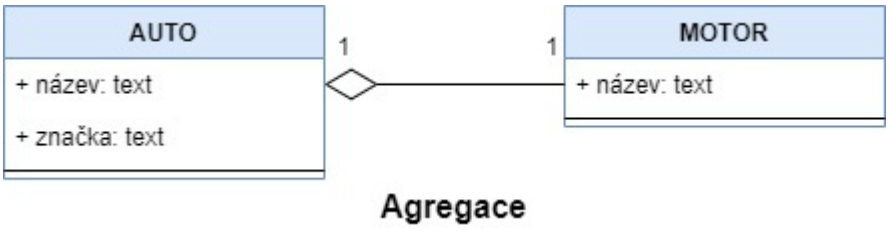
První vazbou je **asociace**, která vyjadřuje přímý vztah mezi jednotlivými třídami. Vhodným příkladem může být třída *Čtenář* a třída *Kniha*.



Obrázek 13: Ukázka asociace

Zdroj: [vlastní zpracování]

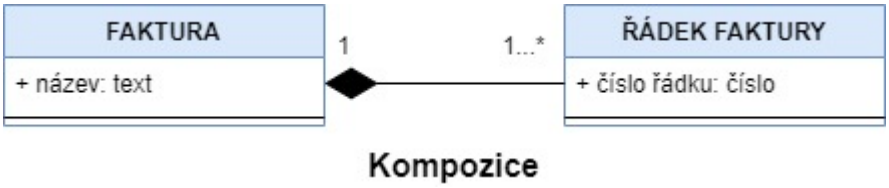
Vazba typu **agregace** říká, že jedna třída je částí třídy druhé, tedy představuje vztah celku k jedné části. Pro lepší pochopení definice této vazby uvedu příklad, kdy agregací by se označil vztah mezi třídou *Auto* a třídou *Motor auta*.



Obrázek 14: Ukázka agregace

Zdroj: [vlastní zpracování]

Poslední vazbou mezi třídami je **kompozice**, která je speciálním typem vazby agregace. Tato vazba propojuje třídy, které nemohou bez sebe samostatně existovat. Odpovídajícím příkladem kompozice je vztah mezi třídou *Faktura* a třídou *Řádek faktury*. (Kanisová, 2012)

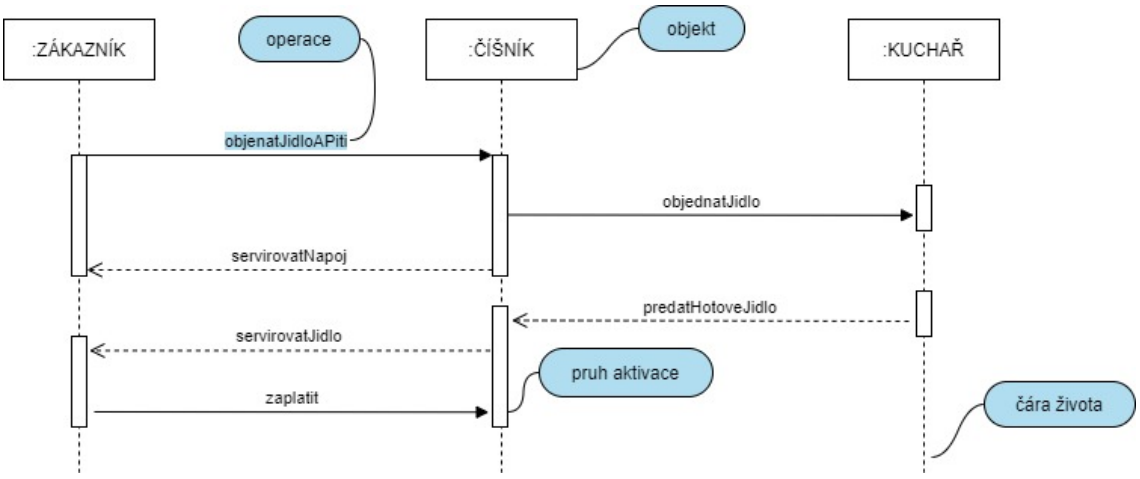


Obrázek 15: Ukázka kompozice

Zdroj: [vlastní zpracování]

SEKVENČNÍ DIAGRAM

Sekvenční diagram znázorňuje chování konkrétního scénáře a zobrazuje předávání jednotlivých zpráv mezi objekty v rámci daného případu užití. Obecně navazuje na scénář a popisuje, jak se daný systém bude chovat, právě z hlediska vybraného scénáře. Sekvenční diagram se vyjadřuje pomocí **objektů**, s kterými je spjata **čára života** a pruh aktivace a dále **zpráv**, které si mezi sebou objekty posílají. **Pruh aktivace** představuje okamžik, ve kterém daný objekt odesílá nebo přijímá zprávu. Jako ukázka sekvenčního diagramu je uveden příklad *Objednání jídla a pití v restauraci*. (Fowler, 2019)



Obrázek 16:Ukázka sekvenčního diagramu

Zdroj: [vlastní zpracování]

1.5.2 TECHNOLOGICKÁ ÚROVEŇ

V konceptuální úrovni se tedy uživatelské požadavky umisťují do určitého modelu, který popisuje základní funkčnost systému. V této úrovni se vytvořený model rozšiřuje o operace. Dále je Class diagram transformován do relací pomocí vazeb, které byly určeny v konceptuální úrovni a následně je provedena normalizace pro odstranění anomálií z modelu.

OPERACE V CLASS DIAGRAMU

V technologické úrovni se do vytvořeného Class diagramu z konceptuální úrovně vytváří takzvané **operace**, pomocí kterých se právě tento diagram výrazně odlišuje od ER diagramu. Operace obecně vyjadřují nějakou činnost, kterou je schopná daná třída provádět a vyjadřuje se pomocí slovesné vazby. Příkladem názvu operace může být *Přidej zákazníka* nebo *Zadej jméno*. Každá operace může mít kromě názvu také jiné vlastnosti, konkrétně **viditelnost**, která určuje, jaké operace mají přístup k atributům. Operace mají tři typy přístupů: (Fowler, 2019)

- Soukromý přístup (značen -) – říká, že k atributu mají přístup pouze ty operace, které jsou využité v příslušné třídě.
- Veřejný přístup (značen +) – určuje, že k atributu mají přístup operace ze všech tříd.

- Chráněný přístup (značen #) – zaručuje, že k atributu mají přístup pouze ty operace, které jsou využité v příslušné třídě nebo ve třídě jejich potomků. (Kanisová, 2012)

1.6 DÍLČÍ SHRNUTÍ

Tato kapitola se věnuje obecnému vysvětlení pojmu databáze a jejích základních vlastností, dále datovému modelování a následně popsání dvou přístupů modelování.

V první řadě je nutné definovat pojem databáze, což je nějaké místo, ve kterém je možné uchovávat data o konkrétním objektu. Tato práce pracuje konkrétně s relační databází, což je určitý druh databáze, která dokáže propojovat jednotlivé tabulky mezi sebou. Tento typ databáze musí splňovat určitá pravidla, a to atomicitu, konzistenci, izolaci a trvalost.

V další části této kapitoly je vysvětlen princip datového modelování a vymezení jeho úrovní, tedy konceptuální, technologické a implementační úrovně. Je zde vysvětleno, že v konceptuální úrovni dochází k vymezení uživatelských požadavků na danou databázi a k vytváření diagramů daného přístupu. Dále je vysvětleno, že v úrovni technologické se již vytvořené diagramy z konceptuální úrovně transformují do návrhových modelů a vzniká tak výsledný relační model dat. V poslední úrovni, tedy v implementační je pak výsledná databáze zavedena do provozu. Tato část kapitoly je zakončena představením čtyř typů datových modelů, konkrétně se jedná o Hierarchický, Síťový, Relační a Objektový datový model.

Poslední částí kapitoly je podrobný popis vytvoření výsledného relačního datového modelu. Začínající fází analýzy a pokračující fází návrhu za použití dvou přístupů modelování, a to Strukturovaného a Objektově – orientovaného přístupu. Oba přístupy jsou popsány v úrovni konceptuální a technologické, aby bylo znázorněno, jak obecně postupovat při návrhu databáze. V této části jsou také popsány jednotlivé diagramy včetně jejich ukázek, které lze v různých fázích využít a také kombinovat pro dokonalý výsledek databáze.

2 VÝBĚR ZAMĚŘENÍ VZOROVÉHO PŘÍKLADU

Modernizace města formou konceptu Smart city mě osobně zaujala natolik že, jsem si toto téma zvolila jako vzorový příklad vytvoření databáze. Je možné, že se toto téma může setkat s kritikou spojenou se změnou něčeho, co jistým způsobem funguje. Podle mého názoru je, ale cesta digitalizace města vhodná, v některých případech i nezbytná. Po seznámení s principy chytrého města jsem si uvědomila, že spoustu těchto „vylepšení“ je součástí mého každodenního života. Příkladem mohou být chytré autobusové zastávky, tedy elektronické informační tabule zobrazující informace o příjezdu daných spojů, včetně zpoždění. Složité vyhledávání spojů pomocí papírového rozpisu a dlouhé někdy marné čekání na spoj, si dnes již nedokážu představit. Dalším příkladem může být také inteligentní veřejné osvětlení. Veřejné osvětlení intenzitu svícení přizpůsobuje svému okolí. Ovlivňuje je denní světlo a pohyb v blízkosti dané lampy.

2.1 SMART CITY

Za Smart City neboli chytré město se považuje město, ve kterém jsou využívány moderní technologie, které vedou ke zlepšení kvality života ve městě. Může se jednat například o chytré parkování, inteligentní veřejné osvětlení, dobíjecí stanice pro elektromobily, chytré autobusové zastávky nebo chytré odpady. (Slavík, 2017)

Starosta města, které se chce stát Smart se snaží vybrat ty vylepšení, která budou přínosná jak pro obyvatele, tak i pro úředníky. Nejvíce by se město mělo rozvíjet v následujících oblastech:

- ICT infrastruktura – jedná se o zvolení vhodné komunikační sítě, ukládání a sdílení dat na cloudu, podpora start-upů nebo například sběr a zpracování dat pomocí IoT (internet věcí). Za internet věcí se považují přístroje, které lze obsluhovat pomocí internetu (Žák, 2017). Ve Smart City se může IoT využít například při řízení dopravního provozu, hlídání úrovně hluku a znečištění vzduchu nebo při odhalení poruch elektřiny, vody apod. (Slavík, 2017)
- Energetika – zde se jedná o snahu využít obnovitelné zdroje, které se přirozeně obnovují, tedy nelze je vyčerpat. Jedná se především o energii větru, slunečního záření a vody. Dalším příkladem rozvinutí energetiky je inteligentní veřejné osvětlení, které je popsané níže. (Slavík, 2017)

- Doprava – rozrůstá se nabídka elektromobilů a hybridních automobilů, které snižují emise a šetří přírodu i obyvatele měst, a zároveň i snižují úroveň hluku. Dále se rozvoje dopravy týče chytré parkování nebo například chytré autobusové zastávky, které jsou opět popsané níže. (Slavík, 2017)
- Odpadové hospodářství – chytré odpady fungují tak, že do popelnice jsou umístěny senzory, které sledují naplněnost popelnic odpadem. Díky tomu nemůže docházet k přeplnění popelnic a znečištění ulic. V některých městech jsou dokonce speciální kontejnery, které mají podzemní trubky a odpad se tak dostane přímo do centrály, kde se dále třídí. (Vintrová, 2019)

Chytré město by obecně mělo zjednodušit život především občanům města, ale také pomůže posunout město na vyšší úroveň a zajistí větší zájem turistů i stěhování lidí do tohoto města.

2.1.1 CHYTRÉ PARKOVÁNÍ

Chytré parkování umožňuje řidičům mít přehled o volných parkovacích místech ve městě jednak přes aplikaci nebo z elektronických tabulí, pomocí senzorů, které jsou zabudované pod parkovacími místy. (ELKO EP, 2016)

Dalším přínosem chytrého parkování je mobilní aplikace, ve které si uživatelé mohou zakoupit parkovací lístek bez toho, aniž by museli dojít k automatu. Parkovací lístek tedy neumisťují za okno a mají ho pouze v mobilním telefonu. Hlídka, která kontroluje parkovací lístky pak jen nahraje registrační značku konkrétního vozidla do chytrého telefonu a zkontroluje, zda má vozidlo zaplacené parkování. (Doseděl, 2020)

2.1.2 INTELIGENTNÍ VEŘEJNÉ OSVĚTLENÍ

Veřejné osvětlení zvyšuje bezpečnost města, může snížit počet dopravních nehod i zmírnit kriminalitu v nočních ulicích. (Slavík, 2017)

Inteligentní osvětlení se rozsvěcuje podle intenzity venkovního světla, to znamená, že pouliční lampy začnou svítit až když se začne stmívat a až se bude rozednívat tak zhasnou. Některé lampy dokážou dokonce šetřit energii takovým způsobem, že mají senzory, díky kterým se rozsvítí až v případě blízkosti lidí. (Poljakov)

2.1.3 CHYTRÉ AUTOBUSOVÉ ZASTÁVKY

Chytré autobusové zastávky slouží k zobrazení informací o příjezdu daného spoje, popřípadě jeho zpoždění. Dále tyto zastávky také umožňují dobíjet mobilní telefon přes USB nebo nabízejí bezplatné připojení k Wi-Fi. (Poljakov)

2.2 DÍLČÍ SHRNUTÍ

Tato kapitola se obecně věnuje pojmu Smart City. Je zde vysvětleno, že pomocí Smart City lze usnadnit život obyvatel města, a zároveň zvýšit jeho úroveň. Důležité je uvědomit si, že koncept Smart City lze využít v několika různých oblastech, které určuje starosta města, právě podle aktuální situace. Může se tedy jednat o oblasti ICT infrastruktury, energetiky, dopravy nebo odpadového hospodářství.

V další části této kapitoly jsou popsány vybrané oblasti Smart City, které se ve městech velmi často používají. První oblastí je chytré parkování, kterému se z části také věnuje tato práce, a to při návrhu databáze pro aplikaci chytrého parkování. Obecně chytré parkování umožňuje přehled o obsazenosti parkovišť a možnost zakoupení parkovacího lístku přes internet.

Další oblastí je inteligentní osvětlení, které pomáhá udržovat bezpečnost města. Princip tohoto osvětlení je v automatickém rozsvícení lampy podle intenzity venkovního světla nebo reakcí na senzor pohybu. Poslední popsanou oblastí jsou chytré autobusové zastávky, které usnadňují cestování především poskytováním informací o příjezdu spojů.

3 NÁVRH POSTUPU PRO VZOROVÝ PŘÍKLAD

V této kapitole jsou uvedeny jednotlivé kroky, které jsou potřebné pro vytvoření výsledné databáze. Tyto kroky jsou také zobrazeny v myšlenkové mapě na konci této kapitoly.

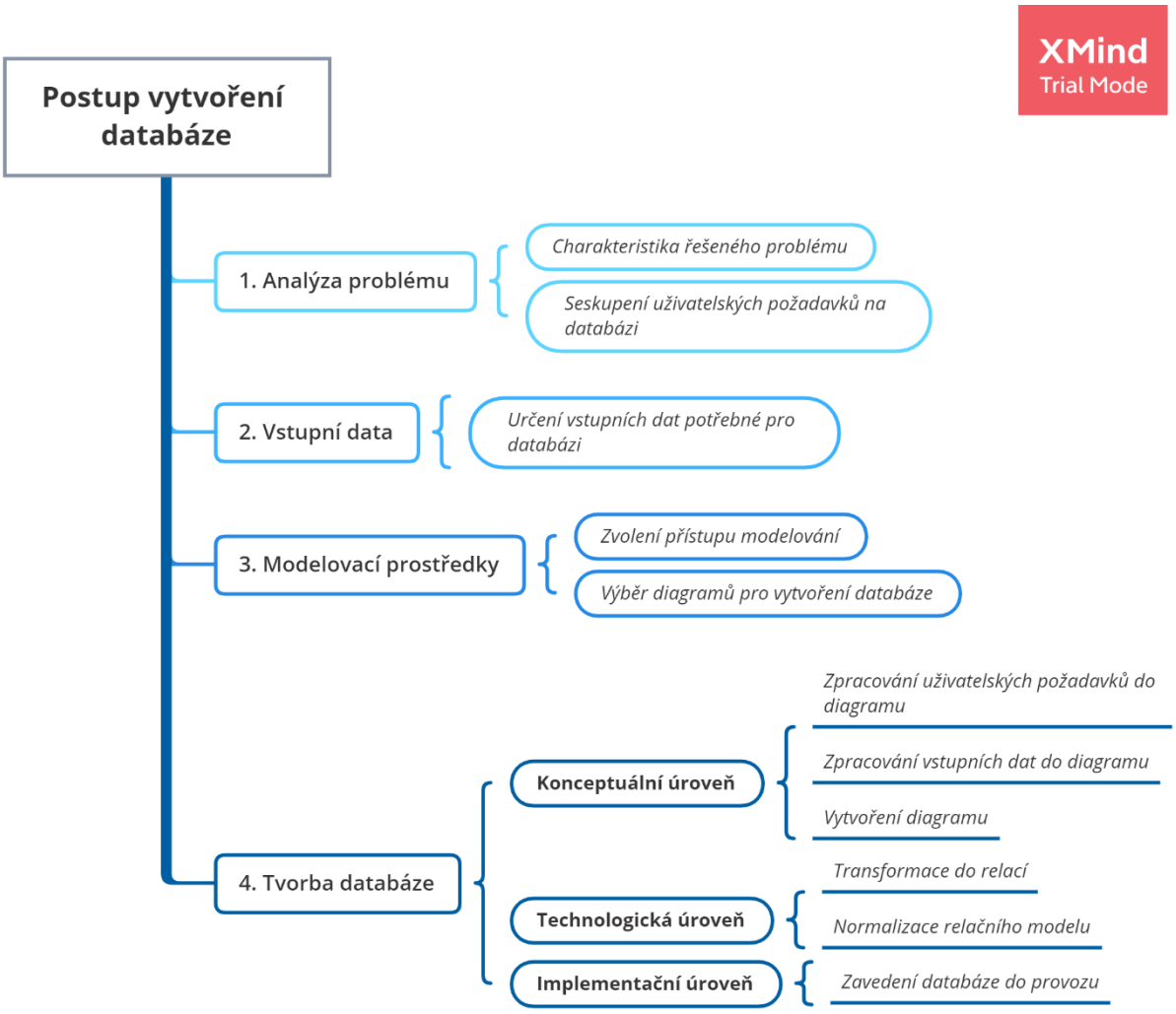
Jako první krok je **Analýza problému**, kde je nutné komunikovat se zájemcem o aplikaci a zjistit tak nezbytné informace pro vytvoření databáze. V tomto kroku je důležité uvést:

- Charakteristiku řešeného problému – v této části je vhodné odpovědět na otázky typu: *„Proč je potřeba navrhnout danou databázi?“* nebo *„V čem může vytvořená databáze pomoci?“*
- Seskupení uživatelských požadavků na databázi – kde je nutné od zájemce zjistit, jak by výsledná databáze měla fungovat.

Druhým krokem jsou **Vstupní data**. Zde je opět nezbytná komunikace se zájemcem o aplikaci, a to z hlediska určení vstupních dat potřebných pro databázi. Zájemce o aplikaci tedy určí, co by daná databáze měla ukládat.

Třetím krokem je určení **Modelovacích prostředků**. V tomto kroku je zapotřebí zvolit přístup modelování, tedy výběr mezi Strukturovaným nebo Objektově – orientovaným přístupem modelování, popřípadě jejich kombinace. Podle určení přístupu modelování jsou následně zvoleny diagramy, ve kterých bude databáze vytvořena.

Posledním krokem je již samotná **Tvorba databáze** v jednotlivých úrovních. V konceptuální úrovni se zpracovávají uživatelské požadavky a vstupní data do zvoleného diagramu. Výstupem z této fáze je vytvořený diagram, který bude následně transformován do relačního datového modelu. V následující úrovni, tedy v úrovni technologické, je již zmíněná transformace do relačního datového modelu a následná normalizace. Výstupem z této fáze je výsledná databáze, která je v implementační úrovni zavedena do provozu.



Obrázek 17: Postup vytvoření databáze

Zdroj: [vlastní zpracování]

4 CHARAKTERISTIKA ŘEŠENÉHO PROBLÉMU

Jako vzorový příklad pro vytvoření databáze jsem se rozhodla využít reálnou problematiku, se kterou se mohou potýkat města, respektive jejich vedení, například starostové měst. Důvodem pro vytvoření databáze a následné aplikace chytrého parkování může být zvýšení úrovně daného města, ale také usnadnění parkování ve vybraném městě. Následující odstavec představuje úvahu, která může být důvodem pro zájem o vytvoření databáze a následné aplikace chytrého parkování u vedení měst.

Určitě každý řidič někdy zažil situaci, kdy projížděl několik minut celé parkoviště a nenašel žádné volné místo pro zaparkování svého auta. Tato situace se běžně stává i mně, když chci ve městě zaparkovat. V dnešní době problém s přeplněnými parkovišti narůstá. Proč tomu tak je? Dřív měla celá rodina jen jedno auto, a ne všichni členové rodiny měli řidičský průkaz. Dnes je tomu ale jinak, v rodině je více aut, která jsou často využívána všemi členy rodiny pro cestu do práce, do školy a na další místa, na které je nutné se dopravit autem. Tyto případy neodsuzuji, i v mé rodině je intenzivně používáno více aut jednotlivými členy rodiny. Nekonečné projíždění parkovištěm s cílem nalezení volného parkovacího místa, je mi tedy velmi blízké. Této ztrátě času je možné předejít zavedením chytrého parkování. Toto řešení informuje řidiče o volných parkovacích místech formou elektronických tabulí nebo mobilních aplikací.

Dalším důvodem pro zavedení chytrého parkování může být velká výhoda elektronického placení parkování, které by nahradilo cestu k parkovacímu automatu a zpět k vozidlu, pro umístění parkovacího lístku. Výhodou elektronického lístku může být také jeho jednoduché časové prodloužení.

5 VSTUPNÍ DATA PRO VYTVOŘENÍ DATABÁZE CHYTRÉHO PARKOVÁNÍ

Navržená databáze by měla evidovat zákazníky, kteří si v aplikaci chytrého parkování chtějí zakoupit parkovací lístek. Tato aplikace může usnadnit parkování spoustě řidičům, kteří již nebudou muset vystupovat ze svých automobilů a hledat parkovací automat, kde si koupí parkovací lístek, který musí umístit za okno svého auta. Zjednoduší také prodloužení platnosti parkovacího lístku – vše bude možné provést mobilní aplikací.

Inspirací pro zvolení dat byla aplikace Smart4City, která v některých městech již funguje.

Níže uvedená data poskytují potřebné informace od zájemce o aplikaci, v případě města se jedná o starostu města, který má zájem o vytvoření databáze pro aplikaci chytrého parkování. S těmito vstupními daty se pak dále pracuje v kapitole 7.

5.1 UŽIVATEL

Uživatel je entita, která uchovává informace o účastnících chytrého parkování. Systém vygeneruje *ID uživatele*, které zajistí jednoznačnou identifikaci uživatelů. Dále uživatel zadá své *jméno*, *heslo* a *email*, na který se mu po dokončení registrace pošle aktivační email. Pomocí emailu a hesla se uživatel přihlašuje do aplikace chytrého parkování. Po registraci a úspěšném přihlášení zadá uživatel *číslo platební karty*, aby mohl platit parkování přes aplikaci. Každý uživatel může mít i více přidaných platebních karet.

5.2 VOZIDLO

Další důležitou entitou je **Vozidlo**, kde se ukládají informace o automobilu nebo například motorovém vozidle uživatele. Uživatel zadává *typ* svého vozidla, zda se jedná o osobní vůz, motorový vůz či nákladní vozidlo (je možné evidovat i více vozidel). Ke každému vozu je nutné pak doplnit i jeho *RZ*, aby bylo možné využít službu chytrého parkování.

5.3 ZPŮSOB PARKOVÁNÍ

Entita **Způsob parkování** slouží ke zjištění, o jaký *typ parkování* (parkování za závorou, parkování na ulici) se jedná. Jako způsob parkování je myšlen způsob vjezdu

na parkoviště. Existují dva typy, a to parkoviště za závorou a parkoviště na ulici, tedy bez závory.

Pokud se jedná o parkování za závorou musí si uživatel vzít parkovací lístek, jak je zvykem i bez aplikace a chytré parkování může využít jen v případě, že přesáhne čas a je potřeba zaplatit poplatek. Zákazník nemusí chodit k automatu na zaplacení parkovného, ale stačí ho zaplatit přes aplikaci chytrého parkování.

V případě, že se jedná o parkování na ulici má zákazník možnost nákup lístku i prodloužení parkovného přes mobilní telefon.

5.4 PLATBA

Před dokončením objednávky je důležitá entita **Platba**. Vygeneruje se *číslo platby* a *částka*, kterou musí zákazník zaplatit za vybrané parkovací místo. Uživatel má možnost přes aplikaci chytrého parkování zjistit jaká *platební metoda* se mu nabízí a vybrat si tu která mu nejvíce vyhovuje (platba mobilní aplikací, hotovostní platba nebo platba kartou).

5.5 PARKOVIŠTĚ

Po otevření aplikace chytrého parkování si uživatel vybere **Parkoviště**. Zobrazí se mu *název* parkoviště a *adresa*, kde se nachází. Dále může uživatel vidět kolik *volných míst* se na parkovišti vyskytuje a díky tomu si vybere, kde své auto zaparkuje. Také se uživateli v aplikaci zobrazí *poznámka*, kde je uvedeno za kolik peněz a na jak dlouho zde může parkovat.

5.6 PARKOVACÍ MÍSTO

U entity **Parkovací místo** se vygeneruje *číslo parkovacího místa*. Důležitým atributem je zde *obsazenost* (volné, obsazené nebo rezervované), kde si uživatel může zobrazit, jestli je možnost zde zaparkovat. Dalším atributem je *datum a čas parkování*, kde si uživatel zadá na jak dlouho bude na místě parkovat a aplikace v telefonu ho pak upozorní několik minut před vypršením parkovného. Dále je zde uvedena *cena* za parkování a *typ parkovacího místa*, tedy zda jde o veřejné stání, rezervováno ZTP, rezervováno MÚ nebo rezervováno pro ostatní.

5.7 DÍLČÍ SHRNUTÍ

Tato kapitola se věnuje upřesnění dat, která jsou potřebná pro vytvoření databáze chytrého parkování. V první části kapitoly je uveden důvod pro vytvoření aplikace, která

umožňuje řidičům usnadnit parkování. Dále je zde uvedeno, že data potřebná pro vytvoření dané databáze jsou inspirována z již vytvořené aplikace pro chytré parkování. Vytvořená databáze má tedy ukázat princip tvoření těchto databází na reálném využití. V další části kapitoly jsou již upřesněná data, která se dále využijí v následující kapitole.

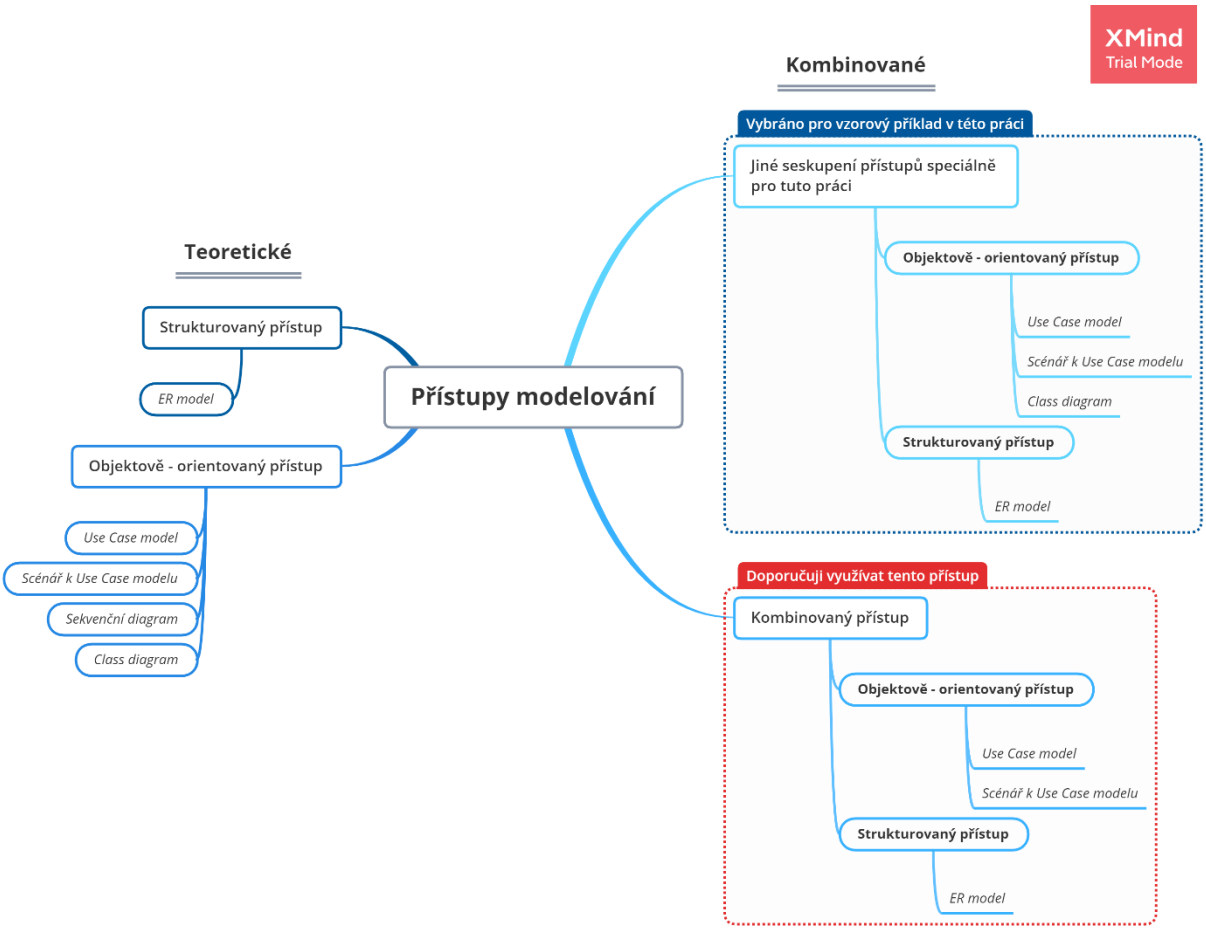
6 VÝBĚR MODELOVACÍCH PROSTŘEDKŮ PRO DATABÁZI CHYTRÉHO PARKOVÁNÍ

K vytvoření databáze pro aplikaci chytrého parkování jsem se rozhodla využít kombinaci dvou přístupů modelování, a to Strukturovaného a Objektově – orientovaného. Důvodem mého rozhodnutí byla jednoduchost struktury ER modelu pro vytvoření databáze a má osobní preference tohoto modelu, kvůli jeho snadnému vytvoření. Za předpokladu, že vytvořená databáze bude dále sloužit pro aplikaci, je vhodné využít také Objektově – orientovaný přístup, který umožňuje zpracovávat objekty podle specifických potřeb daných aplikací.

Rozhodla jsem se tedy v konceptuální úrovni využít Use Case diagram, včetně scénáře z Objektově – orientovaného přístupu modelování, pomocí nichž jsem upřesnila, jak bude daná aplikace fungovat. Byly upřesněny role jednotlivých uživatelů aplikace a funkčnost daného systému. Dále byla databáze tvořena pomocí Strukturovaného přístupu modelování, a to za použití ER diagramu v konceptuální úrovni a následně relačního modelu v úrovni technologické.

Konkrétně pro tento příklad jsem se rozhodla využít také Class diagram z Objektově – orientovaného přístupu pro ukázku demonstrující možnost vytvoření databáze také pomocí tohoto diagramu.

Následující obrázek zobrazuje možné způsoby přístupů modelování pro vytvoření výsledné databáze.



Obrázek 18: Možné způsoby přístupů modelování

Zdroj: [vlastní zpracování]

7 TVORBA DATABÁZE CHYTRÉHO PARKOVÁNÍ V KONCEPTUÁLNÍ ÚROVNI

V této a následující kapitole je zpracován vzorový příklad pro navržení databáze chytrého parkování. Konkrétně v této kapitole je znázorněn postup pro vytvoření databáze v konceptuální úrovni. Prvním krokem je vytvoření Use Case modelu, a to z důvodu následné návaznosti aplikace. Dále je tento model rozšířen o scénáře, které konkrétně popisují, jak bude daná aplikace fungovat. Posledním krokem v konceptuální úrovni je vytvoření ER modelu, ve kterém jsou uložena vstupní data popsána v kapitole 5. Tyto data jsou vložena do entit, které jsou mezi sebou vzájemně propojeny a tvoří tak finální ER model.

V této fázi, tedy v konceptuální úrovni, se zpracovávají požadavky od zájemce o databázi. V případě chytrého parkování se jedná o vedení měst.

7.1 USE CASE MODEL

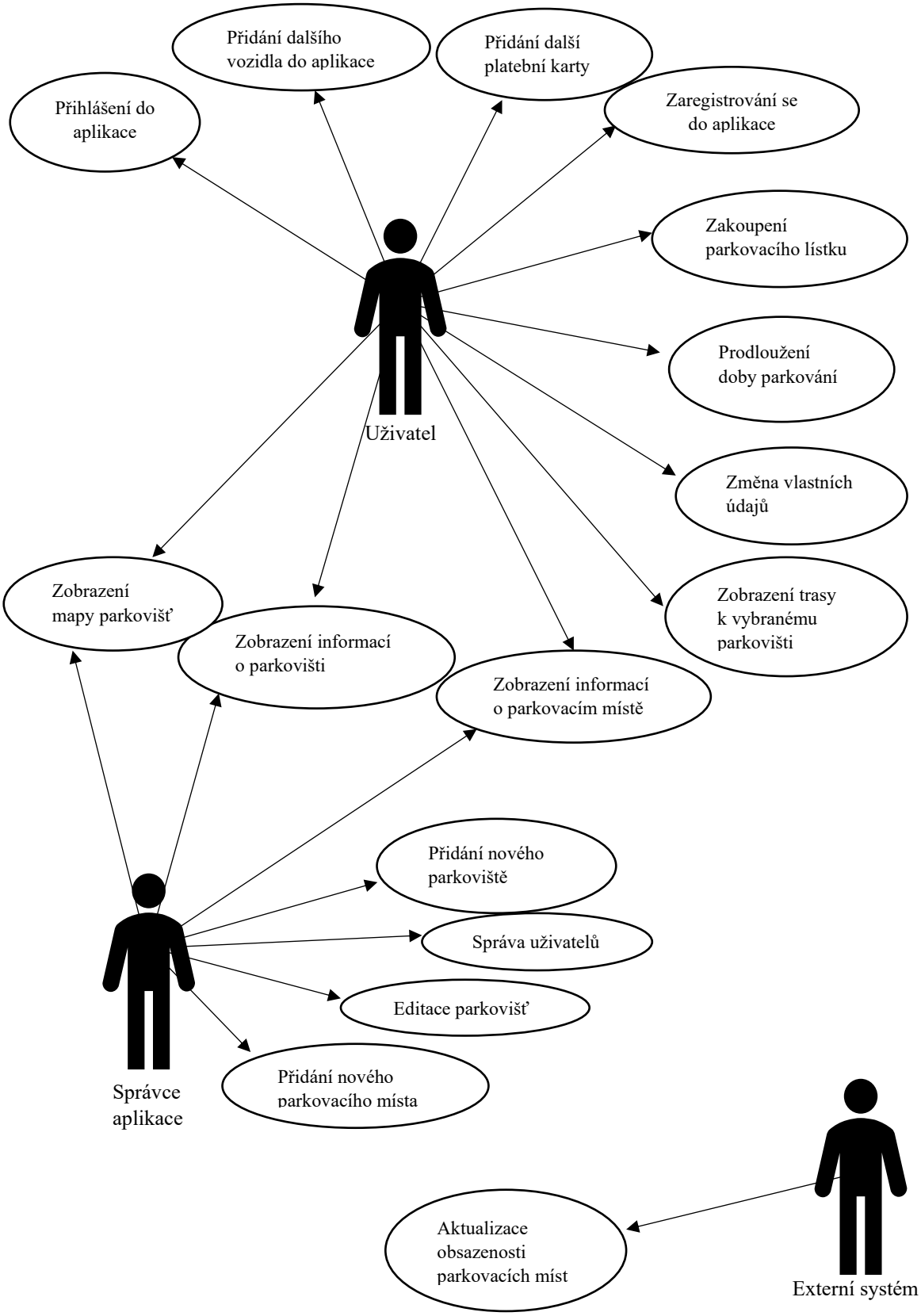
Use Case model specifikuje funkcionalitu budoucí aplikace. Tento model je také vhodné využít pro upřesnění si požadavků na samotnou databázi. Jak již bylo zmíněno v jiné kapitole Use Case se skládá z aktérů a případů užití. Aktér představuje osobu nebo například systém, který komunikuje s vytvořeným systémem a případ užití představuje jednotlivé činnosti, které je možné v tomto systému vykonávat. (Kanisová, 2012)

V Use Case modelu pro databázi chytrého parkování byli určeni tři aktéři a šestnáct případů užití.

Prvním aktérem je **Uživatel** aplikace, což je osoba, která se rozhodne využívat tuto aplikaci. K tomuto uživateli jsou přiřazeny jednotlivé případy užití, tedy oprávnění, která určují, jaké činnosti může tento aktér v aplikaci provádět. Tento uživatel má tedy možnost *zaregistrovat se* a následně se *přihlásit do aplikace*, *přidat vozidla* a *platební karty do aplikace*, dále si také může *změnit své údaje*, které má zadané v aplikaci. Při výběru parkoviště pro zaparkování svého auta si uživatel aplikace může zobrazit *mapu všech parkovišť*, *informace o konkrétním parkovišti* a o *konkrétním parkovacím místě* a následně také *trasu k vybranému parkovišti*. V neposlední řadě má tento aktér možnost *zakoupení parkovacího lístku* a také *prodloužení parkovací doby*.

Dalším aktérem v Use Case modelu je **Správce aplikace**, který celou databázi a aplikaci spravuje. Konkrétně má oproti uživateli aplikace možnost *přidat nové parkoviště* a *nové parkovací místo* do evidence parkovišť a parkovacích míst v aplikaci. Dále také může dané *parkoviště editovat*, tedy upravovat informace například o ceně parkování v konkrétním parkovišti. Posledním specifickým právem je u tohoto uživatele *správa uživatelů* aplikace. Správce aplikace má také možnost *zobrazit si mapu parkovišť* ve městě, dále *informace o konkrétním parkovišti* a *parkovacím místě*.

Posledním aktérem v Use Case modelu pro databázi chytrého parkování je **Externí systém**. Tento aktér provádí pravidelnou aktualizaci obsazenosti všech parkovišť ve městě. Díky tomu se uživatelovi aplikace zobrazuje počet volných míst ve vybraném parkovišti.



Obrázek 19: Use Case model

Zdroj: [vlastní zpracování]

7.2 VYBRANÉ SCÉNÁŘE K USE CASE

V této části kapitoly se vytvářejí scénáře k Use Case modelu pro databázi chytrého parkování. Pro ukázku jsou vybrány pouze tři případy, a to **Zaregistrování se do aplikace**, **Přidání nového parkoviště** a **Přidání nového parkovacího místa**. Tyto scénáře konkrétně popisují jednotlivé činnosti systému.

Při vytváření samotného scénáře, k již vytvořenému modelu Use Case je nejprve důležité zadat *Název Use Case*, který představuje název daného případu užití. Dalším prvkem ve scénáři je *Popis*, kde je vhodné uvést co daný scénář vyjadřuje. Důležitými prvky ve scénáři jsou také *Počáteční* a *Koncový stav*, kde je vyjádřen počátek a závěr scénáře. Posledním prvkem před tvorbou hlavního a alternativního scénáře je *Spouštěcí událost*, ve které je vhodné uvést, jakou činností je daný scénář spuštěn. Další částí scénáře je již samotný *Hlavní scénář*, ve kterém jsou uvedeny jednotlivé kroky včetně aktéra a jeho činností, které vedou k úspěšnému dokončení scénáře, tedy k jeho *Konečnému stavu*. U každého scénáře je nutné vytvořit také *Alternativní scénář*, který provádí jiné řešení kroků v případě neúspěšného provedení hlavního scénáře. (Kanisová, 2012)

Důležité je psát scénáře s podrobnými informacemi, tedy s jednotlivými integritními omezeními. Za integritní omezení se považují datové typy, omezení velikosti, tedy například kolik znaků je možné zadat do jednotlivých polí nebo další nezbytné informace. Tyto informace je důležité zadat z důvodu možnosti práce více osob na této databázi nebo následné vytvoření navazující aplikace. Je tedy důležité, aby bylo z informací ze scénáře zřejmé, jak má daná databáze fungovat i pro osobu, která tento scénář nevytvářela.

Název Use Case	**Zaregistrování se do aplikace**
Aktér	Uživatel
Popis	Postup, jak se může uživatel zaregistrovat do aplikace chytrého parkování
Počáteční stav	Uživatel se chce zaregistrovat do aplikace
Konečný stav	Úspěšná registrace nového uživatele
Spouštěcí událost	Spuštění aplikace chytrého parkování
Hlavní scénář	
Krok 1:	A: Spustí aplikaci chytrého parkování
Krok 2:	S: Zobrazí možnost přihlášení, kde se zobrazí tlačítko *Registrace*, na které má možnost kliknout neregistrovaný uživatel
Krok 3:	A: Klikne na tlačítko *Registrace*
Krok 4:	S: Zobrazí formulář pro registraci nového uživatele
Krok 5:	A: Vyplní pole formuláře: jméno – text (20) email – text (20)

	heslo – text (20) typ vozidla – text (10) RZ – text (8) číslo platební karty – číslo (16) platnost platební karty – datum a čas CVC/CVV – číslo (3)
Krok 6:	A: Klikne na tlačítko *Zaregistrovat*, které se nachází na konci formuláře
Krok 7:	S: Přiřadí nově zaregistrovaného uživatele do seznamu všech zaregistrovaných uživatelů
Alternativní scénář	
Krok 6:	Aktér chybně vyplní údaje do formuláře S: Napíše upozorňující hlášku „Chybně vyplněné údaje“ a zamezí přiřazení uživatele do seznamu uživatelů A: Opraví chybně vyplněné údaje
Krok 6:	Aktér nevyplní všechny údaje do formuláře S: Napíše upozorňující hlášku „Nevyplněné pole formuláře“ a zamezí přiřazení uživatele do seznamu uživatelů A: Vyplní prázdné pole ve formuláři

Tabulka 2: Scénář k Use Case – Zaregistrování se do aplikace

Zdroj: [vlastní zpracování]

Název Use Case	**Přidání nového parkoviště**
Aktér	Správce aplikace
Popis	Postup, jak může správce aplikace chytrého parkování přidat nové parkoviště do evidence parkovišť
Počáteční stav	Správce aplikace chce přidat nové parkoviště do systému
Konečný stav	Úspěšné přidání nového parkoviště
Spouštěcí událost	Spuštění aplikace chytrého parkování
Hlavní scénář	
Krok 1:	A: Spustí aplikaci chytrého parkování
Krok 2:	S: Zobrazí možnost přihlášení, kde se zobrazí pole *Email* a *Heslo*
Krok 3:	A: Zadá email a heslo do těchto polí a klikne na tlačítko *Přihlásit* (správce aplikace má speciální email a heslo pro možnost úpravy aplikace)
Krok 4:	S: Zobrazí domovskou stránku správce aplikace, kde je na výběr ze 7 tlačítek, na které může kliknout: Zobrazení mapy parkovišť Zobrazení informací o parkovišti Zobrazení informací o parkovacím místě Přidání nového parkoviště Správa uživatelů Editace parkovišť Přidání nového parkovacího místa
Krok 5:	A: Klikne na tlačítko *Přidání nového parkoviště*
Krok 6:	S: Zobrazí formulář pro přidání parkoviště
Krok 7:	A: Vyplní pole formuláře: Název – text (20) Adresa – text (50) Počet volných míst – číslo (3) Poznámku pro uživatele – text (50) - například lze napsat cenu a dobu parkování

Krok 8:	A: Klikne na tlačítko *Přidat parkoviště*, které se nachází na konci formuláře
Krok 9:	S: Přiřadí nově vytvořené parkoviště do seznamu všech parkovišť ve městě
Alternativní scénář	
Krok 3:	Aktér zadá chybně přihlašovací údaje S: Napíše upozorňující hlášku „Váš přihlašovací email nebo heslo je chybné“ a zamezí přihlášení uživatele do aplikace
Krok 8:	Aktér chybně vyplní údaje do formuláře S: Napíše upozorňující hlášku „Chybně vyplněné údaje“ a zamezí přiřazení parkoviště do seznamu parkovišť A: Opraví chybně vyplněné údaje
Krok 8:	Aktér nevyplní všechny údaje do formuláře S: Napíše upozorňující hlášku „Nevyplněné pole formuláře“ a zamezí přiřazení parkoviště do seznamu parkovišť A: Vyplní prázdné pole ve formuláři

Tabulka 3: Scénář k Use Case – Přidání nového parkoviště

Zdroj: [vlastní zpracování]

Název Use Case	**Přidání nového parkovacího místa**
Aktér	Správce aplikace
Popis	Postup, jak může správce aplikace chytrého parkování přidat nové parkovací místo do vybraného parkoviště
Počáteční stav	Správce aplikace chce přidat nové parkovací místo do systému
Konečný stav	Úspěšné přidání nového parkovacího místa do vybraného parkoviště
Spouštěcí událost	Spuštění aplikace chytrého parkování
Hlavní scénář	
Krok 1:	A: Spustí aplikaci chytrého parkování
Krok 2:	S: Zobrazí možnost přihlášení, kde se zobrazí pole *Email* a *Heslo*
Krok 3:	A: Zadá email a heslo a klikne na tlačítko *Přihlásit* (správce aplikace má speciální email a heslo pro možnost úpravy aplikace)
Krok 4:	S: Zobrazí domovskou stránku správce aplikace, kde je na výběr ze 7 tlačítek, na které může kliknout: Zobrazení mapy parkovišť Zobrazení informací o parkovišti Zobrazení informací o parkovacím místě Přidání nového parkoviště Správa uživatelů Editace parkovišť Přidání nového parkovacího místa
Krok 5:	A: Klikne na tlačítko *Přidání nového parkovacího místa*
Krok 6:	S: Zobrazí seznam všech evidovaných parkovišť
Krok 7:	A: Vybere parkoviště, ke kterému chce přidat nové parkovací místo a klikne na jeho název
Krok 8:	S: Zobrazí informace o vybraném parkovišti: Název – text (20) Adresa – text (50) Počet volných míst – číslo (3) Poznámku pro uživatele – text (50) - například cenu a dobu parkování
Krok 9:	A: Klikne na tlačítko *Zobrazit parkovací místa*
Krok 10:	S: Zobrazí seznam všech (volných, rezervovaných i obsazených) parkovacích míst u daného parkoviště

Krok 11:	A: Klikne na tlačítko *Přidat nové parkovací místo*
Krok 12:	S: Zobrazí formulář pro přidání parkovacího místa
Krok 13:	A: Vyplní pole formuláře: Cena – měna Typ parkovacího místa – výběr z domény (veřejné stání, rezervováno ZTP, rezervováno MÚ nebo rezervováno pro ostatní)
Krok 14:	A: Klikne na tlačítko *Přidat parkovací místo*, které se nachází na konci formuláře
Krok 15:	S: Přiřadí nově vytvořené parkovací místo, mezi již evidovaná parkovací místa u daného parkoviště
Alternativní scénář	
Krok 3:	Aktér zadá chybně přihlašovací údaje S: Napíše upozorňující hlášku „Váš přihlašovací email nebo heslo je chybné“ a zamezí přihlášení uživatele do aplikace
Krok 14:	Aktér chybně vyplní údaje do formuláře S: Napíše upozorňující hlášku „Chybně vyplněné údaje“ a zamezí přiřazení parkovacího místa do evidence parkovacích míst u daného parkoviště A: Opraví chybně vyplněné údaje
Krok 14:	Aktér nevyplní všechny údaje do formuláře S: Napíše upozorňující hlášku „Nevyplněné pole formuláře“ a zamezí přiřazení parkovacího místa do evidence parkovacích míst u daného parkoviště A: Vyplní prázdné pole ve formuláři

Tabulka 4: Scénář k Use Case – Přidání nového parkovacího místa

Zdroj: [vlastní zpracování]

7.3 ER MODEL

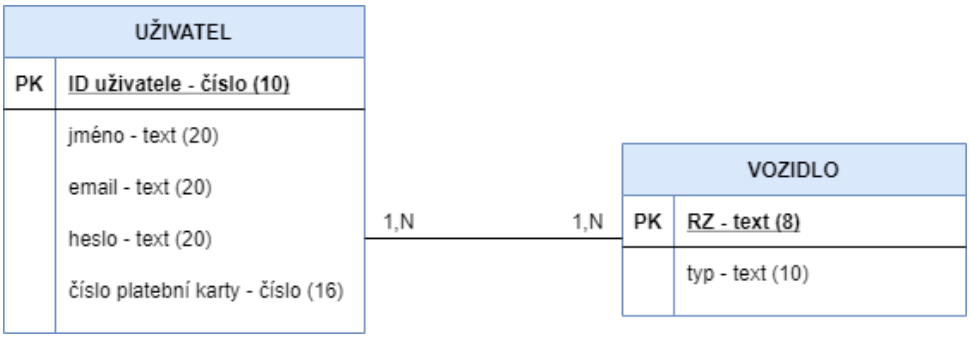
ER model převádí vstupní data do jednotlivých entit a přiřazuje jim atributy a datové typy. Tyto entity jsou následně propojené pomocí vazeb, které jsou níže zobrazené u entit pro databázi chytrého parkování.

Uživatel – Vozidlo

Tento vztah určuje, že *Uživatel* musí vlastnit jedno nebo více vozidel, tomu odpovídá vazba 1,N. Příslušné *Vozidlo* musí vlastnit alespoň jeden uživatel, ale může ho vlastnit i další, například člen rodiny nebo další zaměstnanec, pokud se jedná o firemní auto. V tomto případě se jedná tedy opět o vazbu 1,N.

U entity *Uživatel* je jako primární klíč (PK), určen atribut *ID uživatele*, kterému je přiřazen datový typ číslo. Dalšími atributy v této entitě jsou *jméno, email* a *heslo*, kterým je přidělen datový typ text. Poslednímu atributu *číslo platební karty* odpovídá datový typ číslo.

U entity *Vozidlo* je atribut *RZ* určen jako primární klíč (PK) a je mu přiřazen datový typ text. Tento datový typ je také přidělen dalšímu atributu, a to atributu *typ*.



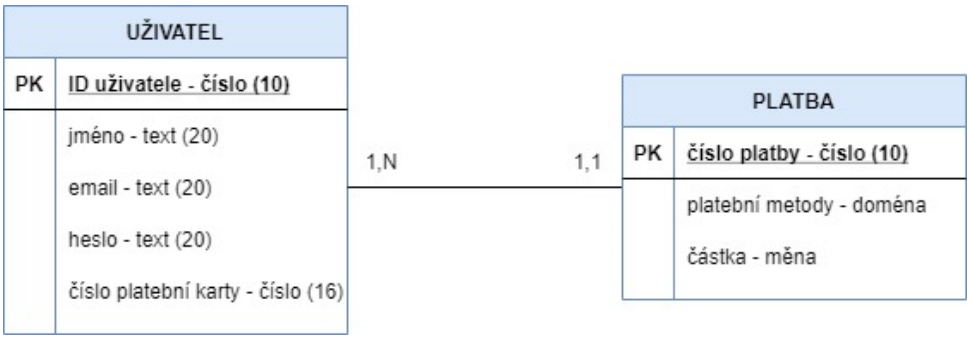
Obrázek 20: Vztah mezi entitami Uživatel – Vozidlo

Zdroj: [vlastní zpracování]

Uživatel – Platba

Tento vztah vyjadřuje, že *Uživatel* musí provést jednu nebo více plateb v této aplikaci, jedná se tedy o vazbu 1,N. Určitá *Platba* se, ale váže pouze k jednomu uživateli. To znamená, že příslušná platba musí být provedena a má právo ji provést pouze jeden konkrétní uživatel, proto vazba 1,1.

Primární klíč a atributy u entity *Uživatel* jsou popsány ve vztahu Uživatel – Vozidlo. Primárním klíčem (PK) u entity *Platba* je *číslo platby*, které je přiřazen datový typ číslo. Dalším atributem v této entitě je *platební metoda*, které je přidělen datový typ doména, ve které je možné vybrat ze tří způsobů platby. Poslední atribut je *částka*, které odpovídá datový typ měna.



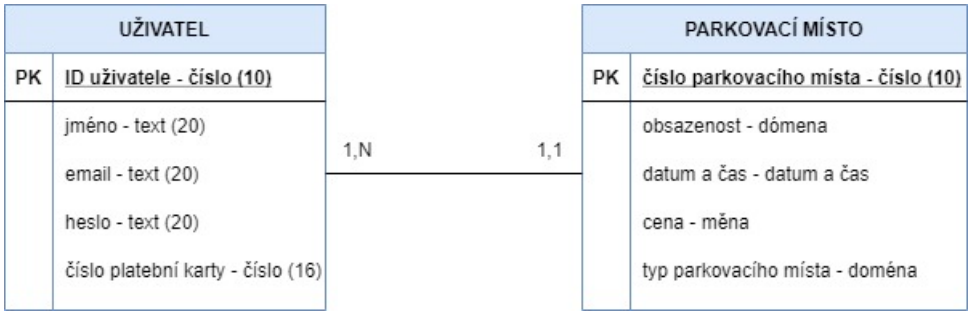
Obrázek 21: Vztah mezi entitami Uživatel – Platba

Zdroj: [vlastní zpracování]

Uživatel – Parkovací místo

Z tohoto vztahu vyplývá, že *Uživatel* si musí zaplatit jedno nebo i více parkovacích míst, například pro své další vozidlo, tomu odpovídá vazba 1,N. Dané *Parkovací místo*, ale patří v daný okamžik jen jednomu zákazníkovi, proto je zde vazba 1,1.

Primární klíč a atributy u entity *Uživatel* jsou popsány ve vztahu Uživatel – Vozidlo. U entity Parkovací místo je jako primární klíč (PK) určen atribut *číslo parkovacího místa*, kterému odpovídá datový typ číslo. Dalším atributem je *obsazenost*, který určuje, zda je konkrétní parkovací místo volné, obsazené nebo rezervované, proto je zde datový typ doména. U atributu *datum a čas* je použit datový typ datum a čas a u atributu *cena* je datový typ měna. Atributu *typ parkovacího místa* odpovídá datový typ doména, u kterého je na výběr ze čtyř možností.



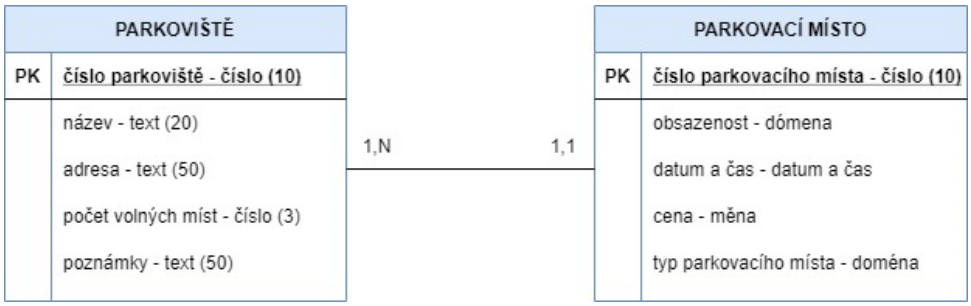
Obrázek 22: Vztah mezi entitami Uživatel – Parkovací místo

Zdroj: [vlastní zpracování]

Parkoviště – Parkovací místo

Tento vztah říká, že každé *Parkoviště* musí obsahovat jedno nebo více parkovacích míst, proto vztah 1,N. Zato určité *Parkovací místo* musí být vázáno pouze k jednomu danému parkovišti, tedy nemůže být zároveň i na jiném parkovišti. Tomuto případu odpovídá vazba 1,1.

U entity *Parkoviště* je jako primární klíč (PK) určen atribut *číslo parkoviště*, kterému je přidělen datový typ číslo. Dalšími atributy v této entitě jsou *název, adresa, počet volných míst* a *poznámky*. Těmto atributům odpovídá datový typ text, kromě *atributu počet volných míst*, kterému je přiřazen datový typ číslo. Primární klíč a atributy u entity *Parkovací místo* jsou popsány ve vztahu Uživatel – Parkovací místo.



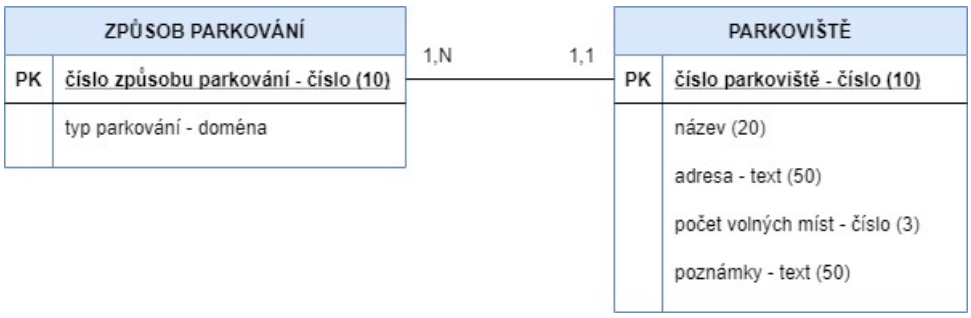
Obrázek 23: Vztah mezi entitami Parkoviště – Parkovací místo

Zdroj: [vlastní zpracování]

Způsob parkování – Parkoviště

Tento vztah vyjadřuje, že určitý *Způsob parkování* musí být alespoň na jednom parkovišti, to znamená že například parkování za závorou se musí vyskytovat alespoň na jednom parkovišti ve městě, tomu odpovídá vazba 1,N. Vybrané *Parkoviště* musí mít ale jenom jeden způsob parkování, tedy nemůže být jedno parkoviště zároveň za závorou a bez závory, proto vazba 1,1.

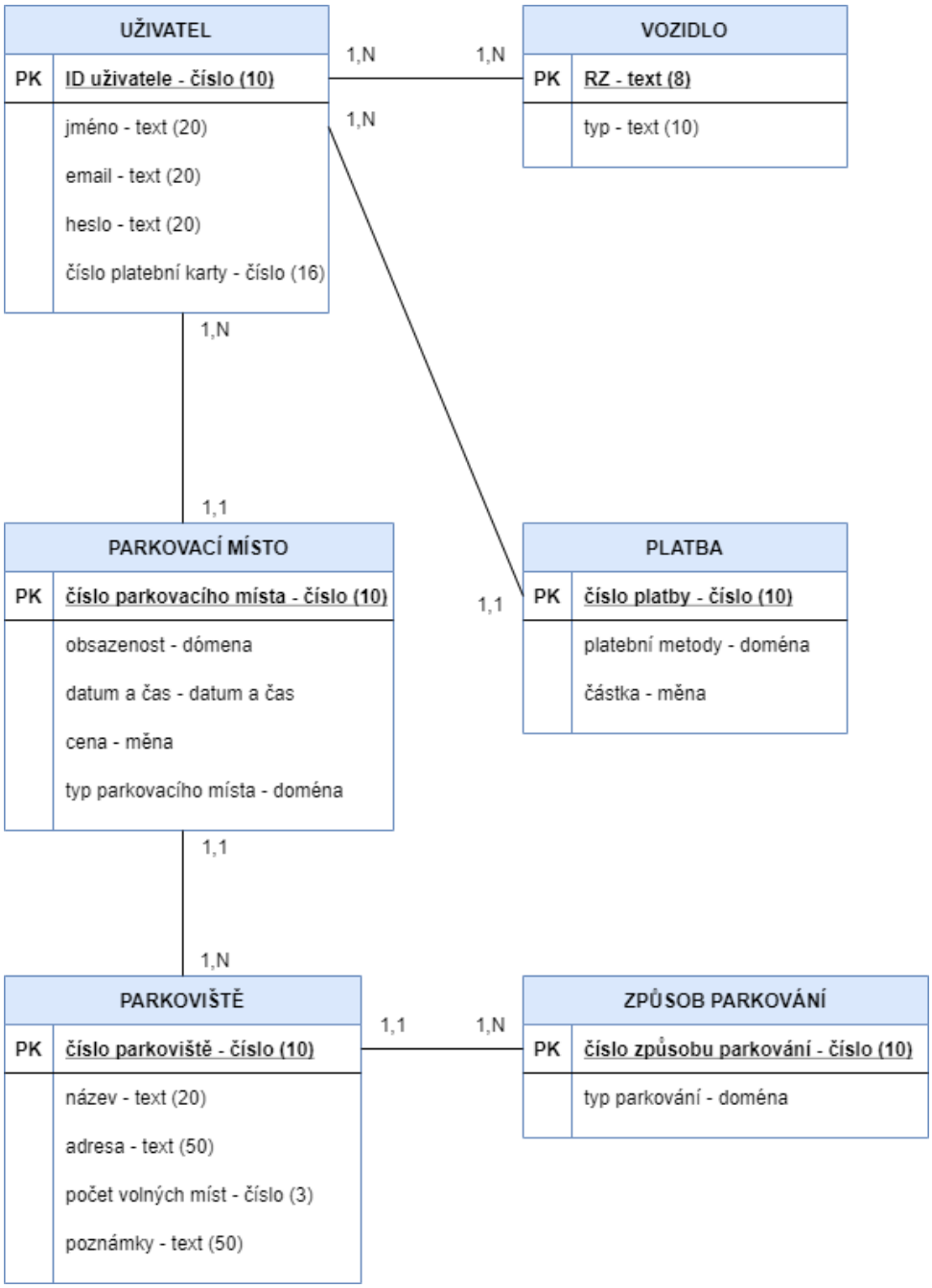
U entity *Způsob parkování* je atribut *číslo způsobu parkování*, který je určen jako primární klíč (PK). Jako další atribut je zde *typ parkování*, kterému je přiřazen datový typ doména, kde je na výběr z možností parkování za závorou a parkování na ulici. Primární klíč a atributy u entity *Parkoviště* jsou popsány ve vztahu Parkoviště – Parkovací místo.



Obrázek 24: Vztah mezi entitami Způsob parkování – Parkoviště

Zdroj: [vlastní zpracování]

Po určení a propojení vazeb mezi jednotlivými entitami vzniká výsledný ER model pro databázi chytrého parkování.



Obrázek 25: ER model

Zdroj: [vlastní zpracování]

7.4 DÍLČÍ SHRNUTÍ

V této kapitole je již vytvořena databáze chytrého parkování v konceptuální úrovni. V první části kapitoly je představen postup tvorby a modely pro vytvoření databáze. Další část kapitoly se věnuje vytvoření prvního modelu v konceptuální úrovni, a to Use Case modelu. Je zde konkrétně popsáno, jak byl daný model vytvořen a z jakých prvků se skládá. Výstupem této

části kapitoly je ukázka vytvořeného Use Case modelu pro aplikaci chytrého parkování. Na tuto část kapitoly navazuje vytvoření scénářů ke třem vybraným případům užití z již vytvořeného Use Case modelu. Opět je zde podrobně popsán obecný postup při vytváření těchto scénářů a výstupem jsou tři vytvořené scénáře případů užití. Poslední částí této kapitoly je vytvoření ER modelu. V této části je zobrazeno převedení vstupních dat do ER modelu, včetně určení vazeb mezi jednotlivými entitami. Výstupem je zhotovený ER model pro databázi chytrého parkování.

8 TVORBA DATABÁZE CHYTRÉHO PARKOVÁNÍ V TECHNOLOGICKÉ ÚROVNI

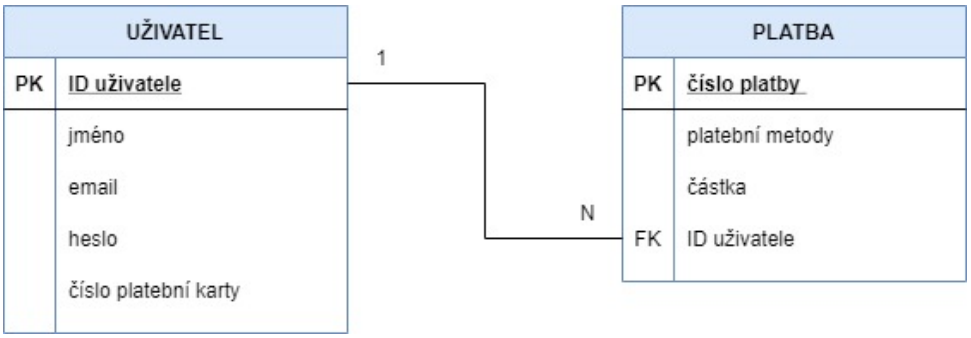
V této kapitole je znázorněn postup pro vytvoření databáze v technologické úrovni. Prvním krokem je transformace vytvořeného ER modelu do modelu relačního. Dojde tedy k vytvoření relací a vazeb mezi nimi. Důležitým krokem je zde také normalizace relačního modelu. V každém modelu se mohou vyskytovat anomálie, tedy chyby, které nedokáže software realizovat a je nutné provést normalizaci. Po provedení normalizace je vytvořen finální relační model, který představuje výslednou databázi chytrého parkování. Tuto databázi je následně možné převést do implementační úrovně, která umožní zavést výslednou databázi do provozu.

8.1 TRANSFORMACE DO RELAČNÍHO MODELU

Vstupem do technologické úrovně je ER model, který se transformuje do relačního modelu. Během transformace vznikají cizí klíče, relace a vazby mezi nimi (Kaluža, 2012). Tyto relace jsou níže zobrazeny při vytváření databáze chytrého parkování.

Uživatel – Platba

Tento vztah měl v ER modelu vazbu u entity *Uživatel* 1,N a u entity *Platba* vazbu 1,1, proto zde vzniknou dvě relace a v entitě *Platba*, u které je vazba 1,1 se vytvoří cizí klíč (FK) z primárního klíče (PK) v entitě *Uživatel*. Tyto dva obsahově stejné klíče (primární a cizí) se následně propojí. Relace, kde se nachází cizí klíč, tedy relace *Platba* se pak označuje vazbou N a relace *Uživatel*, ve které je primární klíč vazbou 1.

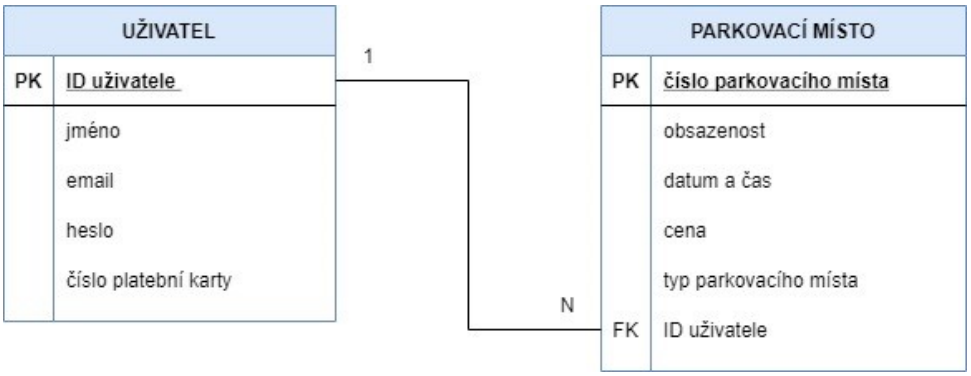


Obrázek 26: Relace Uživatel – Platba

Zdroj: [vlastní zpracování]

Uživatel – Parkovací místo

U tohoto vztahu v ER modelu je opět vazba 1,N a 1,1. Konkrétně u entity *Uživatel* je vazba 1,N a u entity *Parkovací místo* je vazba 1,1. Vytvoří se tedy dvě relace, kde v entitě *Parkovací místo*, která má vazbu 1,1 se vytvoří cizí klíč (FK) z primárního klíče (PK), který se nachází v entitě *Uživatel*. Opět se tyto dva klíče (primární a cizí) následně propojí. Nakonec relace, ve které se vyskytuje cizí klíč, což je relace *Parkovací místo* se pak označuje vazbou N a relace *Uživatel*, která obsahuje klíč primární se označuje vazbou 1.

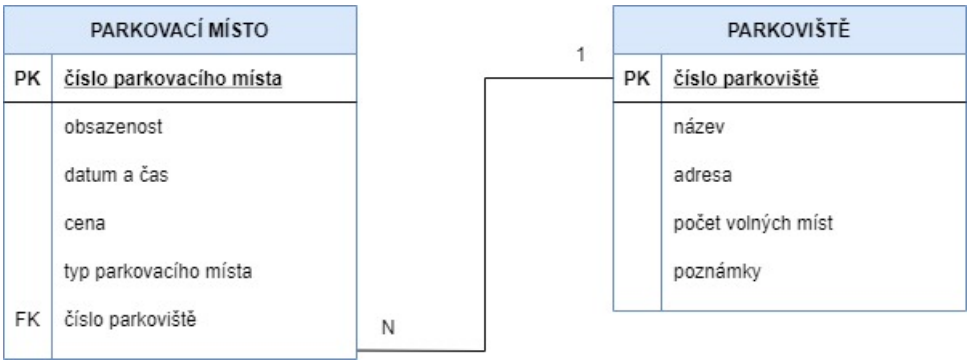


Obrázek 27: Relace Uživatel – Parkovací místo

Zdroj: [vlastní zpracování]

Parkovací místo – Parkoviště

U tohoto vztahu v ER modelu je opět vazba 1,N a 1,1. Jedná se o entitu *Parkovací místo*, u které je vazba 1,1 a u entity *Parkoviště* se nachází vazba 1,N. Jsou tedy vytvořeny dvě relace. Konkrétně v relaci *Parkovací místo*, která má u entity vazbu 1,1 je vložen cizí klíč (FK) z primárního klíče (PK) v entitě *Parkoviště*. Tyto dva klíče jsou následně opět propojené a u relace *Parkovací místo*, ve které je obsažen cizí klíč vznikne vazba značená N a u relace *Parkoviště*, která obsahuje primární klíč vznikne vazba značená 1.

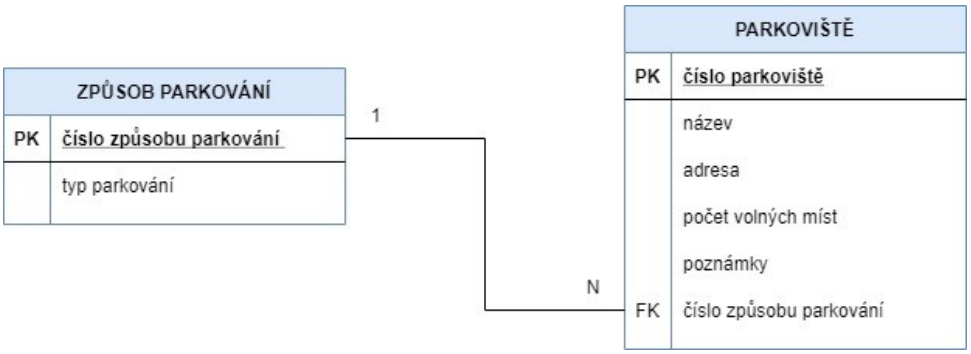


Obrázek 28: Relace Parkovací místo – Parkoviště

Zdroj: [vlastní zpracování]

Způsob parkování – Parkoviště

V tomto vztahu v ER modelu je opět vazba 1,N a 1,1. Entita *Způsob parkování* je označená vazbou 1,N a entita *Parkoviště* vazbou 1,1. Opět se tedy vytvoří dvě relace, kde v relaci *Parkoviště*, která byla označená vazbou 1,1 vznikne cizí klíč (FK), který je převzat z primárního klíče (PK) v entitě *Způsob parkování*. Primární a cizí klíč je následně propojen a u relace *Způsob parkování*, kde se nachází primární klíč vznikne vazba značená 1 a u relace *Parkoviště*, ve které se nachází klíč cizí vznikne vazba značená N.



Obrázek 29: Relace Způsob parkování – Parkoviště

Zdroj: [vlastní zpracování]

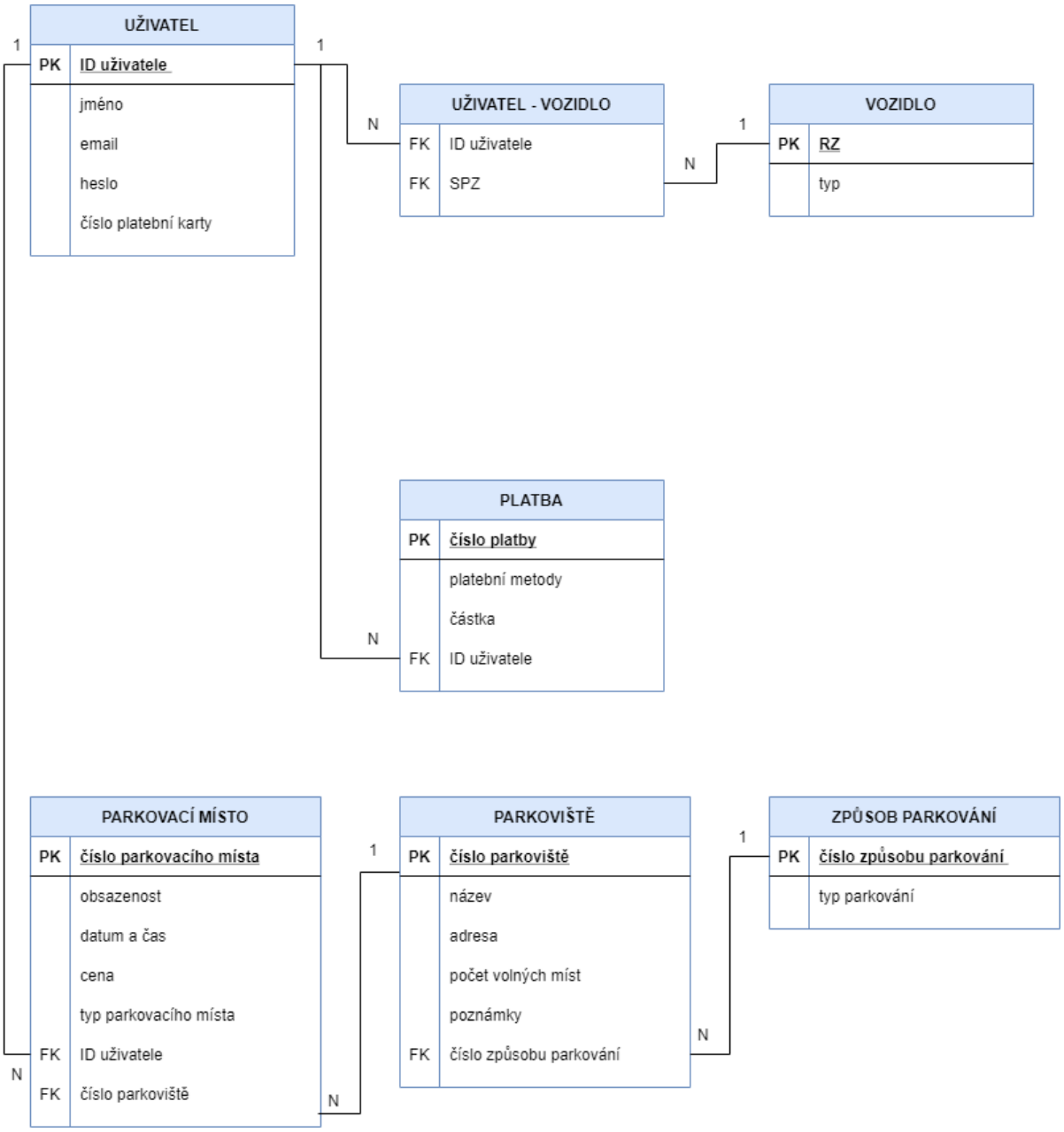
Uživatel – Vozidlo

U tohoto vztahu v ER modelu v konceptuální úrovni se nevyskytuje žádná entita s vazbou 1,1, proto vzniknou tři relace, z nichž třetí relace je vazební. Tato relace je pojmenována jako *Uživatel-Vozidlo*. V této relaci pak vzniknou cizí klíče, které jsou převzaty z primárních klíčů (PK) ze dvou zbylých relací (relace *Uživatel* a relace *Vozidlo*). Opět se propojí obsahově stejné klíče (primární a cizí). Následně relace *Uživatel-Vozidlo*, ve které se nacházejí cizí klíče (FK) se označí vazbou N a relace s primárním klíčem, tedy relace *Uživatel* a relace *Vozidlo* se označí vazbou 1.

Obrázek 30: Relace Uživatel – Vozidlo

Zdroj: [vlastní zpracování]

Po vytvoření relací a jejich vazeb vzniká relační model pro databázi chytrého parkování.



Obrázek 31: Relační model

Zdroj: [vlastní zpracování]

8.2 NORMALIZACE RELAČNÍHO MODELU

Jak již bylo zmíněno před uvedením databáze do provozu je nutné vytvořený relační model normalizovat a odstranit tak z daného modelu anomálie.

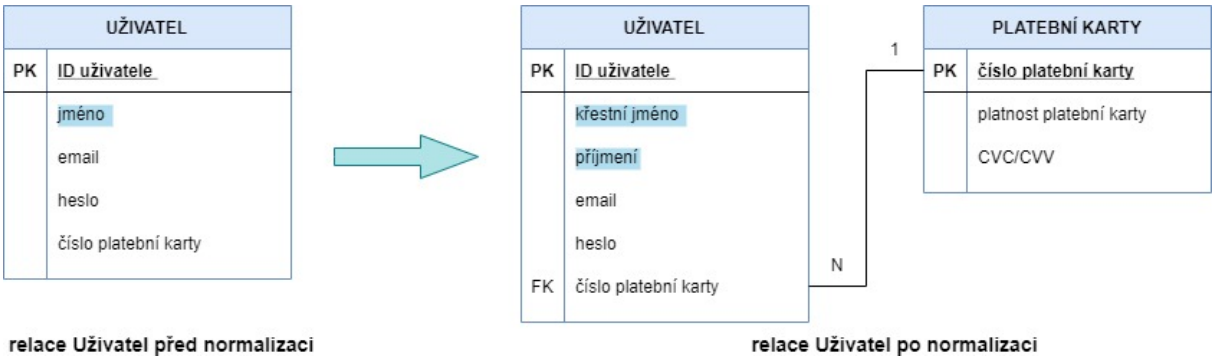
Existuje řada normálních forem, ale v praxi se využívají především první tři normální formy. V relaci *Uživatel* a v relaci *Parkoviště* se vyskytují anomálie, které se odhalí pomocí první normální formy (1NF) a třetí normální formy (3NF).

Relace Uživatel

První anomálie v relaci *Uživatel* je vícehodnotový atribut a odhalí se pomocí první normální formy (1NF). Jedná se o atribut *jméno*, které lze rozdělit na *křestní jméno* a *příjmení.*

Druhou anomálií v této relaci je závislost neklíčového atributu i na dalších neklíčových atributech, která se odhalí pomocí třetí normální formy (3NF). Jedná se o atribut *číslo platební karty* a je tedy nutné ho umístit do nové relace.

Do nově vytvořené relace *Platební karty* jsou přidány atributy *číslo platební karty*, *platnost platební karty* a její *CVC/CVV*, kde *číslo platební karty* je primárním klíčem (PK).

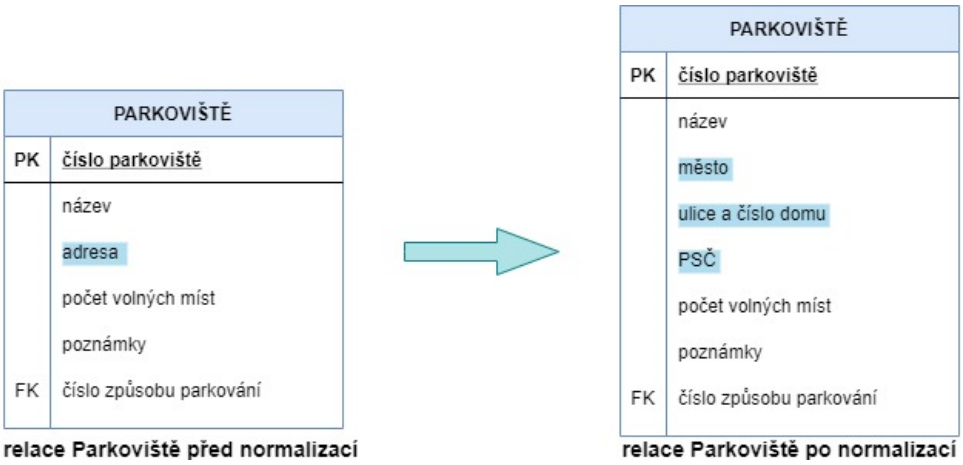


Obrázek 32: Relace Uživatel před a po normalizaci

Zdroj: [vlastní zpracování]

Relace Parkoviště

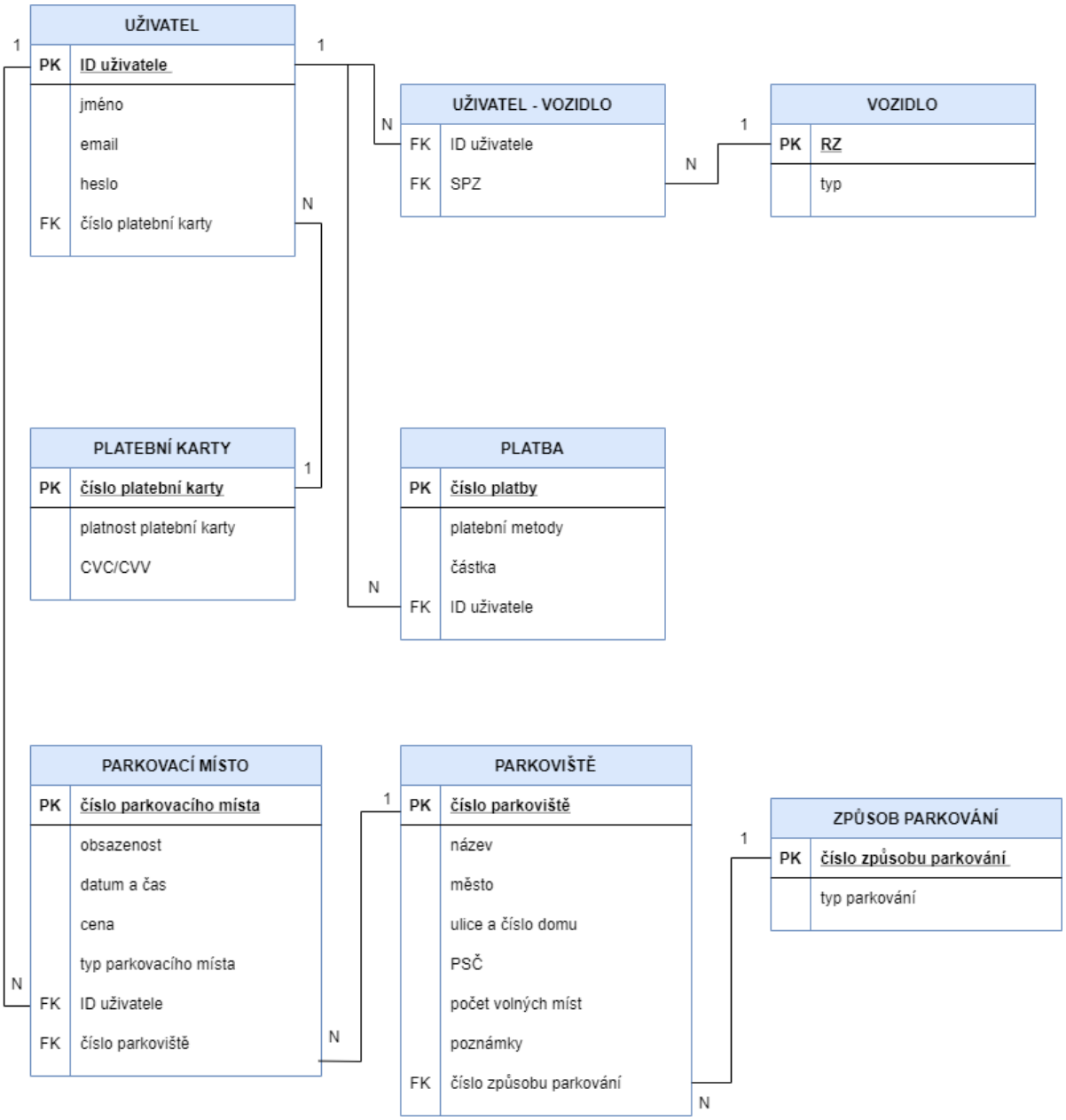
U relace *Parkoviště* je anomálie odhalena pomocí první normální formy (1NF) a jedná se o vícehodnotový atribut *adresa*, který se může rozdělit na atribut *město, ulice a číslo domu* a na *PSČ*.



Obrázek 33: Relace Parkoviště před a po normalizaci

Zdroj: [vlastní zpracování]

Po odhalení anomálií pomocí normálních forem vzniká finální normalizovaný relační model, který je software schopen realizovat a je tedy možné převést databázi chytrého parkování do implementační úrovně a uvést ji do provozu.



Obrázek 34: Normalizovaný relační model

Zdroj: [vlastní zpracování]

8.3 DÍLČÍ SHRNUTÍ

V této kapitole je vytvořena databáze chytrého parkování v technologické úrovni. Tato úroveň vytváří již finální databázi. V první části kapitoly je popsán postup tvorby databáze v této úrovni. V další části kapitoly je provedena transformace ER modelu do modelu relačního. Je zde vysvětlen postup tvorby relací a vazeb mezi nimi. Výstupem je vytvořený relační model. Tento model však obsahuje anomálie, proto je nutné provést normalizaci relačního modelu.

Normalizace je prováděna pomocí normálních forem. Po úspěšném provedení je výstupem z technologické úrovně finální relační datový model. Tento model může být již zaveden do provozu jako databáze chytrého parkování.

9 CLASS DIAGRAM

Vzhledem k tomu, že se jedná o vzorový příklad, který je určen pro studenty se zájmem o tvorbu databází, je zde také uvedeno řešení příkladu chytrého parkování pomocí Class diagramu.

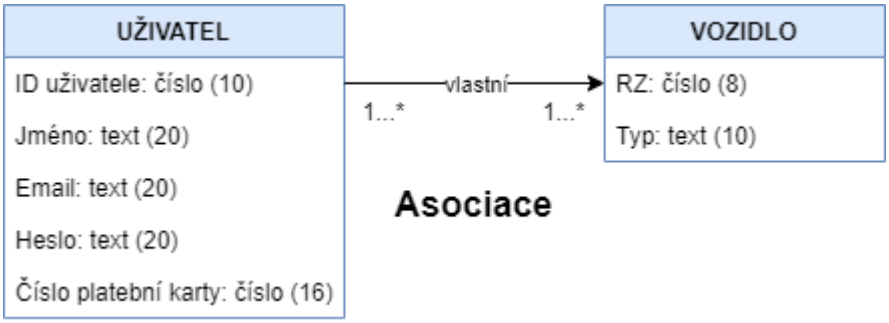
Vstupem do tohoto diagramu jsou opět vstupní data z kapitoly 5. Tyto data jsou v Class diagramu převedeny do tzv. tříd, které jsou mezi sebou propojeny pomocí vazeb. V tomto diagramu je také zapotřebí uvést vztahy na úrovni instance, a to asociace, agregace a kompozice. Dále se v tomto diagramu určují u jednotlivých atributů datové typy. (Kanisová, 2012)

V případě, že je k atributu přiřazen datový typ doména, vznikne z tohoto atributu výčtová třída (<<*enumeration*>>), kde je vypsán seznam možných hodnot atributů (Urban, 2003). Výčtová třída se týká atributů Obsazenost, Typ parkovacího místa, Platební metody a Typ parkování.

Uživatel – Vozidlo

Tento vztah říká, že *Uživatel* vlastní jedno nebo více vozidel, proto vazba 1…* a příslušné *Vozidlo* vlastní jeden nebo více uživatelů, čemuž opět odpovídá vazba 1…*. Jednotlivé třídy, které se vyskytují v tomto vztahu odpovídají vazbě asociace, jelikož mají mezi sebou navzájem vztah.

Třída *Uživatel* obsahuje atributy *Jméno*, *Email* a *Heslo*, které mají přiřazen datový typ text. Dalšími atributy v této třídě jsou *ID uživatele* a *Číslo platební karty*, kterým odpovídá datový typ číslo. Ve třídě *Vozidlo* jsou uloženy atributy *RZ* a *Typ*, které jsou datového typu text.



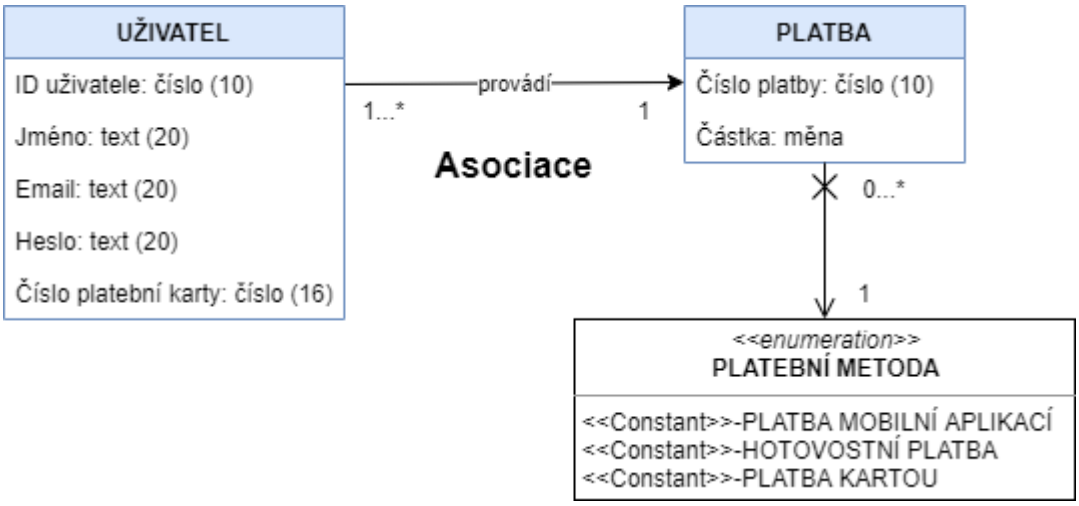
Obrázek 35: Vztah mezi třídami Uživatel – Vozidlo

Zdroj: [vlastní zpracování]

Uživatel – Platba

Tento vztah značí, že *Uživatel* provádí jednu nebo více plateb, proto vazba 1…* a konkrétní *Platbu* provádí pouze jeden uživatel, tomu odpovídá vazba 1. Třídy v tomto vztahu mají mezi sebou vztah, proto je zde přiřazena vazba asociace.

U třídy *Uživatel* jsou atributy určeny v předchozím vztahu. Třída *Platba* obsahuje atribut *Číslo platby*, který má přiřazen datový typ číslo a atribut *Částka*, který je datového typu měna. Atribut *Platební metody* je přiřazen datovému typu doména, proto zde vznikne výčtová třída, ve které je vypsán seznam možných hodnot.



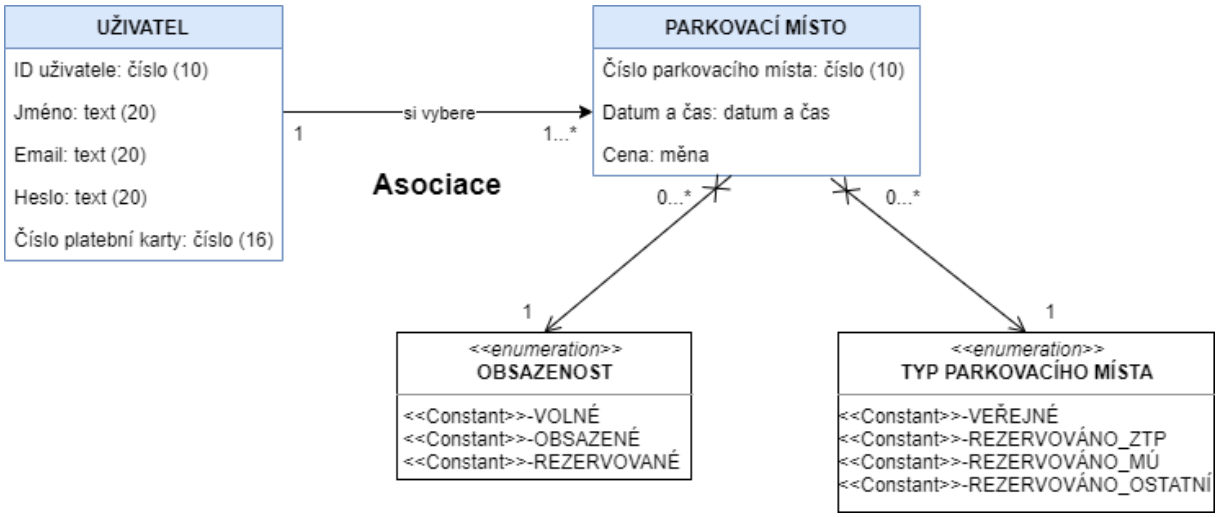
Obrázek 36: Vztah mezi třídami Uživatel – Platby

Zdroj: [vlastní zpracování]

Uživatel – Parkovací místo

Tento vztah vyjadřuje, že *Uživatel* si vybere jednu nebo více parkovacích míst, jedná se tedy o vazbu 1…*. Jedno konkrétní *Parkovací místo* si ale vybere pouze jeden uživatel, proto vztah 1. Opět třídy v tomto vztahu mají mezi sebou vazbu asociace.

Jednotlivé atributy u třídy *Uživatel* jsou popsány u vazby *Uživatel-Vozidlo*. Třída *Parkovací místo* je tvořena atributy *Číslo parkovacího místa*, kterému je přiřazen datový typ číslo, dále atributem *Datum a čas*, kterému odpovídá datový typ datum a čas a atribut *Cena*, kterému je přiřazen datový typ měna. Dalšími atributy v této třídě jsou *Obsazenost* a *Typ parkovacího místa*, které odpovídají datovému typu doména. V tomto vztahu vzniknou tedy dvě výčtové třídy se seznamem možných hodnot.



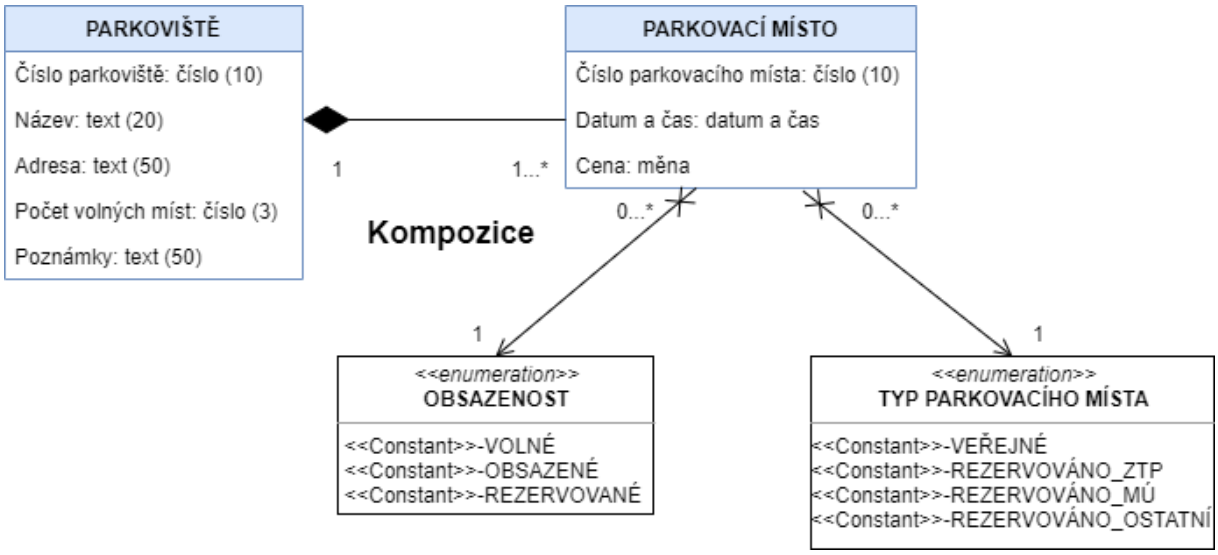
Obrázek 37: Vztah mezi třídami Uživatel – Parkovací místo

Zdroj: [vlastní zpracování]

Parkoviště – Parkovací místo

Z tohoto vztahu vyplývá, že *Parkoviště* obsahuje jedno nebo více parkovacích míst, tomu odpovídá vazba 1…*. Konkrétní *Parkovací místo* obsahuje pouze jedno parkoviště, jedná se tedy o vazbu 1. U tohoto vztahu je určená vazba kompozice, a to z důvodu, že *Parkovací místo* je částí *Parkoviště*, a zároveň nemůže bez něj existovat.

Ve třídě *Parkoviště* jsou obsaženy atributy *Název*, *Adresa* a *Poznámky*, kterým odpovídá datový typ text. Dalšími atributy v této třídě jsou *Číslo parkoviště* a *Počet volných míst*, kterým odpovídá datový typ číslo. Atributy ve třídě *Parkovací místo* jsou popsány ve vztahu *Uživatel–Parkovací místo*.



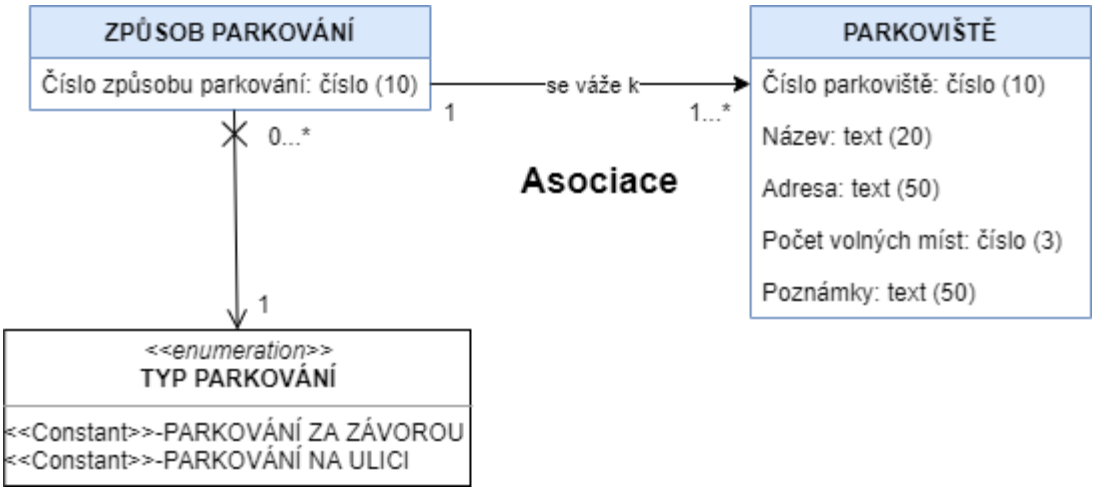
Obrázek 38: Vztah mezi třídami Parkoviště – Parkovací místo

Zdroj: [vlastní zpracování]

Způsob parkování – Parkoviště

Tento vztah určuje, že *Způsob parkování* se váže k jednomu nebo více parkovištím, proto vazba 1…* a *Parkoviště* se váže k jednomu způsobu parkování, tedy vazba 1. Mezi těmito třídami je určitý vztah, proto je zde vazba asociace.

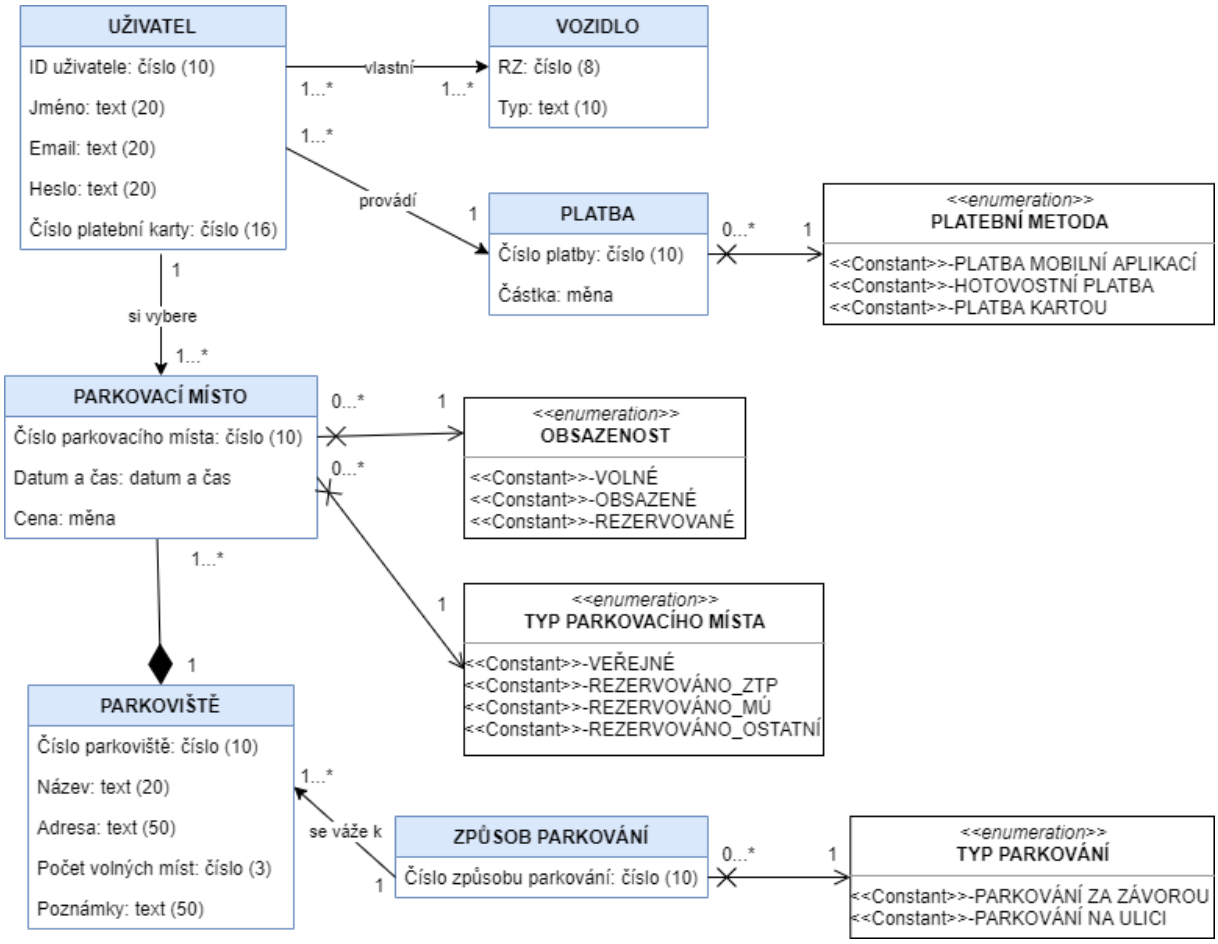
Třída *Způsob parkování* obsahuje dva atributy, a to *Číslo způsobu parkování* a *Typ parkování*. Atributu *Číslo způsobu parkování* odpovídá datový typ číslo a atributu *Typ parkování* je přiřazen datový typ doména. Dojde tedy opět k vytvoření výčtové třídy. Atributy ve třídě *Parkoviště* jsou popsány ve vztahu *Parkoviště-Parkovací místo*.



Obrázek 39: Vztah mezi třídami Způsob parkování – Parkoviště

Zdroj: [vlastní zpracování]

Po určení a propojení vazeb mezi jednotlivými třídami vzniká výsledný Class diagram pro databázi chytrého parkování. Tento diagram by byl následně v technologické úrovni transformován a normalizován a byl by doplněn o příslušné operace. Následně by vznikl výsledný relační datový model, který by mohl být zaveden do provozu.



Obrázek 40: Class diagram

Zdroj: [vlastní zpracování]

9.1 DÍLČÍ SHRNUTÍ

V této kapitole byl vytvořen Class diagram pro vzorový příklad databáze chytrého parkování. Tento diagram byl vytvořen jako další možné řešení příkladu právě pomocí Class diagramu. Podrobně jsou zde popsané jednotlivé vazby mezi třídami, určeny atributy tříd a jejich datové typy. Po určení těchto vazeb je následně vytvořen výsledný Class diagram v konceptuální úrovni. Tento diagram by byl následně v technologické úrovni transformován a normalizován do výsledného relačního datového modelu a zaveden do provozu.

ZAVĚR

Při tvoření této bakalářské práce jsem využila své předešlé znalosti z předmětu Databázové systémy, ale hlavně jsem dostala možnost spojit fakta a vědomosti ze studia, s praktickými situacemi z každodenního života. Uvědomila jsem si důležitost databází a také, jak často jsou používány. Zdokonalení se v oblasti databázových systémů, díky zpracování této práce, pociťuji jako velmi přínosné i v mém začínajícím profesním životě.

Cílem této práce bylo vytvoření přehledného vzorového příkladu tvorby databází, který by měl studentům pomoci pochopit jeho princip. S potěšením mohu prohlásit, že se tento cíl podařilo naplnit, což jsem si ověřila nedávno, kdy mě studentka nižšího ročníku požádala o pomoc s tvorbou databáze. Pro vysvětlení jsem použila postup tvorby, který bakalářská práce popisuje a k mé radosti studentka pochopila princip velmi rychle.

Kromě principu tvorby databází práce popisuje, dva různé přístupy modelování, jednotlivě popsané na začátku této práce. Úrovně pro tyto přístupy byly poté zvlášť popsány a vysvětleny na příkladech.

Další důležitou částí práce je i koncept Smart City, konkrétně jeho část, zaměřená na chytré parkování ve městě, které jsem si, díky hlubšímu pochopení jeho principu, při tvorbě této práce, ještě více oblíbila a v rodném Kolíně parkuji pohodlně už právě jen pomocí aplikace chytrého parkování, které je zde zavedeno již 5. rokem.

Práce také detailně vysvětluje jednotlivé části procesu tvorby databází, kde je každá část vysvětlena tak, aby byla pochopena i studenty, kteří se s tvorbou databází setkávají poprvé. Tato část práce si vyžádala velmi pečlivé hledání informací z mnoha zdrojů a výsledkem je navržená databáze, kterou by bylo možné využít pro aplikaci a byl také vytvořen podrobný, ale přehledný popis celého procesu tvorby databází.

SEZNAM LITERATURY

Co je relační databáze?, 2021. *Oracle Česká republika* [online]. Česká republika [cit. 2021-01-31]. Dostupné z: https://www.oracle.com/cz/database/what-is-a-relational-database/

DANEL, Roman, 2013. *Strukturovaný přístup k analýze a návrhu IS* [online]. Ostrava [cit. 2021-03-28]. Dostupné z: https://homel.vsb.cz/~dan11/aps/texty/Danel%20-%20APS%20-%20Strukturovany%20pristup%20k%20navrhu%20systemu.pdf. Učební text. Vysoká škola báňská - Technická univerzita Ostrava.

Databázové modely, 2010. *Databáze* [online]. [cit. 2021-02-04]. Dostupné z: http://www.databaze.chytrak.cz/modely.htm

DOSEDĚL, Tomáš, 2020. Chytré parkování míří do Pardubic. *SvětChytře.cz* [online]. SocialBooster, s.r.o. [cit. 2020-11-17]. ISSN 2570-8627. Dostupné z: https://www.svetchytre.cz/a/pggjS/chytre-parkovani-miri-do-pardubic

ELKO EP, s.r.o, 2016. Parkování. *Elkoep.cz* [online]. Holešov [cit. 2020-11-17]. Dostupné z: https://www.elkoep.cz/sc-parkovani

FOWLER, Martin, 2019. *Destilované UML*. 3. vydání. Praha: Grada Publishing, a.s. ISBN 978-80-247-2062-3.

KALUŽA, Jindřich a Ludmila KALUŽOVÁ, 2012. *Modelování dat v informačních systémech*. 1. vydání. Praha: Ekopress, s.r.o. ISBN 978-80-86929-81-1.

KANISOVÁ, Hana a Miroslav MÜLLER, 2012. *UML srozumitelně*. 2. vydání. Brno: Computer Press. ISBN 978-80-251-1083-6.

KOPAL, Ondřej, 2015. Relační a nerelační datový model v kontextu business intelligence. *Seriál Svět databází a webová integrace* [online]. [cit. 2021-01-31]. Dostupné z: http://www.web-integration.info/cs/blog/relacni-a-nerelacni-datovy-model-v-kontextu-business-intelligence/

KROENKE, David M. a David J. AUER, 2015. *Databáze*. 1. vydání. Brno: Computer Press. ISBN 978-80-251-4352-0.

KYBKALO, Anatoliy, 2013. *Datové modelování* [online]. Praha [cit. 2021-02-04]. Dostupné z: https://old.unicorncollege.cz/bakalarske-prace/archiv-2013/kybkalo-anatoliy/attachments/BP_-_Kybkalo_Anatoliy.pdf. Bakalářská práce. Unicorn College, Katedra informačních technologií. Vedoucí práce Miroslav Žďárský.

MATULA, Jan, 2015. *Modelování IS – Strukturovaný a objektově orientovaný přístup (UML)* [online]. Brno [cit. 2021-03-28]. Dostupné z: https://is.muni.cz/el/phil/podzim2015/VIKBA18/um/VIKBA18_-_2._blok_ZS_2015.pdf. Učební text. Masarykova univerzita, Fakulta informatiky.

MICROSOFT 365, 2021. Příručka k relacím mezi tabulkami. *Microsoft* [online]. [cit. 2021-02-10]. Dostupné z: https://support.microsoft.com/cs-cz/office/p%C5%99%C3%ADru%C4%8Dka-k-relac%C3%ADm-mezi-tabulkami-30446197-4fbe-457b-b992-2f6fb812b58f

MICROSOFT 365, 2021. Úvod k datovým typům a vlastnostem pole. *Microsoft* [online]. [cit. 2021-02-10]. Dostupné z: https://support.microsoft.com/cs-cz/office/%C3%BAvod-k-datov%C3%BDm-typ%C5%AFm-a-vlastnostem-pole-30ad644f-946c-442e-8bd2-be067361987c

POKORNÝ, Jaroslav a Michal VALENTA, 2020. *Databázové systémy*. 2. přepracované vydání. Praha: Česká technika - nakladatelství ČVUT. ISBN 978-80-01-06696-6.

POLJAKOV, Nikita a Barbora PŮLPÁNOVÁ. *SMART CITY: cesta za lepším životem ve městě* [online]. Praha: Economia a.s. [cit. 2020-11-17]. Dostupné z: http://service.ihned.cz/smartcity/

SKŘIVAN, Jaromír, 2008. *Datové modely a návrhy relačních schémat* [online]. Praha [cit. 2021-02-04]. Dostupné z: https://sites.ff.cuni.cz/uisk/wp-content/uploads/sites/62/2016/01/Datov%C3%A9-modely-a-n%C3%A1vrhy-rela%C4%8Dn%C3%ADch-sch%C3%A9mat_Sk%C5%99ivan.pdf. Učební text. Univerzita Karlova, Filozofická fakulta, Ústav Informačních studií a knihovnictví.

SLAVÍK, Jakub, 2017. *Smart city v praxi: jak pomocí moderních technologií vytvářet město příjemné k životu a přátelské k podnikání*. 1. vydání. Praha: Profi Press s.r.o. ISBN 978-80-86726-80-9.

URBAN, Vít, 2003. *E-learningové materiály pro výuku jazyka UML* [online]. Brno [cit. 2020-11-17]. Dostupné z: https://is.muni.cz/th/255929/fi_b/BP_urban.pdf. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Dalibor Toth.

VÁVRA, David, 2012. Relační databázové systémy. *Posterus* [online]. **5**(5) [cit. 2021-01-31]. ISSN 1338-0087. Dostupné z: https://www.posterus.sk/?p=13526

VINTROVÁ, Kateřina, 2019. *Využití koncepce Smart Cities v kontextu rozvoje města Uherské Hradiště* [online]. Zlín [cit. 2020-11-17]. Dostupné z: https://digilib.k.utb.cz/bitstream/handle/10563/44997/vintrov%c3%a1_2019_dp.pdf?sequence=1&isAllowed=y. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně, Fakulta managementu a ekonomiky. Vedoucí práce Lukáš Danko.

ZELENKA, Petr, 2004. WebML – datové modelování. *WebML* [online]. [cit. 2021-02-04]. Dostupné z: https://www.interval.cz/clanky/webml-datove-modelovani/

ŽÁK, Petr, 2017. Smart Cities - inteligentní města. *Bestdecision.cz* [online]. Praha [cit. 2020-11-17]. Dostupné z: https://www.bestdecision.cz/smart-cities-inteligentni-mesta/

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

RZ – registrační značka vozidla

PSČ – poštovní směrovací číslo

IoT – internet věcí

PK – primární klíč

FK – cizí klíč

NF – normální formy

ID – jedinečný identifikátor