Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Datová integrita v relačně databázovém zpracování

Jan Urban

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Urban

Informatika

Název práce

Datová integrita v relačně databázovém zpracování

Název anglicky

Data integrity in a relational database processing

Cíle práce

Bakalářská práce je tematicky zaměřená na problematiku datové integrity v relačně databázovém zpracování dat. Cílem práce je:
a) vymezit teoretické principy problematiky relačních databází a to především v záležitosti datové integrity,
b) zmapovat současnou úroveň řešené problematiky a identifikovat požadavky na ni kladené,
c) navrhnout a následně ověřit možnosti řešení těchto požadavků na konkrétním příkladu z praxe,
d) ověřené záležitosti včetně jejich přínosů zobecnit pro další možná použití.

Metodika

Použitá metodika zadané bakalářské práce bude založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Stěžejní pro vypracování této závěrečné práce budou metody a techniky relačně databázové technologie v kontextu s problematikou datové integrity. Navrhované řešení bude zohledňovat identifikované požadavky a očekávání spojená s řešenou záležitostí. Na podkladě syntézy teoretických poznatků a dosažených výsledků budou formulovány závěry této bakalářské práce a následně zobecněny pro další možná použití.

Doporučený rozsah práce

40-50 stran

Doporučené zdroje informací

BEGG, C., CONOLLY, T., HOLOWCZAK, R.: Mistrovství databáze, profesionální průvodce tvorbou efektivních databází. Computer Press. Brno 2009. ISSBN 978-80-251-2328-7

BRYLA, B., LONEY, K.: Mistrovství v Oracle Database 10g. Computer Press Brno 2006. EAN 978802512779

HERMANDEZ, M.: Návrh databází, GRADA 2005. ISBN 80-247-0900-7.

LACKO, L: ORACLE. Správa, programování a použití databázového systému. Computer Press Brno 2007. EAN 97880251149002.

LONEY, K.: Oracle Database, kompletní průvodce. Computer press Brno 2010. ISBN 978-80-251-2489-5

MOLINARO, A.: SQL Kuchařka programátora. Computer Press. Brno 2009. ISBN 978-80-251-2617-2

POKORNÝ, J.: Databázové systémy. ČVUT Praha 2013. ISBN 978-80-01-05212-9

Předběžný termín obhajoby

2015/06 (červen)

Vedoucí práce

doc. Dr. Ing. Václav Vostrovský, Ph.D.

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 03. 2015

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Datová integrita v relačně databázovém zpracování" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 16.3.2015 _____

Jan Urban

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Václavu Vostrovskému, Ph.D. za vedení této práce, jeho odborné rady a veškerý čas, který mi věnoval během přípravy bakalářské práce.

Datová integrita v relačně databázovém zpracování

Data integrity in a relational database processing

Souhrn

Bakalářská práce se zabývá problematikou datové integrity v relačně databázovém zpracování. V teoretické části jsou vysvětleny principy relačních databází a pojmy spojené především s datovou integritou a tvorbou integritních omezení. Dále je v této kapitole popsán jazyk SQL.

Druhá část práce je zaměřena na zhodnocení současné situace při navrhování integritních omezení v databázích. Pro demonstraci byl vybrán nový centrální registr vozidel. Je zde úvod do dané problematiky, dále pak vlastní návrh řešení problematiky centrálního registru vozidel za účelem odstranění jeho chyb a nakonec závěrečné zhodnocení řešeného problému.

V poslední části je namodelován příklad na demonstraci správného návrhu integritních omezení. Je zde vytvořen ER diagram a popis jednotlivých entit a atributů. Dále návrh integritních omezení v rámci řešeného příkladu a jejich realizace pomocí příkazů jazyka SQL. Je zde také vysvětleno, jak důležitý je návrh integritních omezení, aby databáze pracovala korektně a předcházelo se chybám.

Klíčová slova: relačně databázová technologie, datová integrita, SQL, triggery, návrh

Summary

Bachelor thesis deals with the topic of data integrity in a relational database processing. The theoretical part explains the principles of relational databases and ideas mainly connected with data integrity and formation integrity constraints. Furthermore, this chapter describes the SQL language.

The second part of bachelor thesis is focused on the evaluation of the present situation in the drafting of integrity constraints in databases. As an examples, the new central register of vehicles was chosen. There is an introduction of the topic, as well as proper draft solutions to the topic of central register of vehicles, in order to eliminate its mistakes. At the end there is valorization of the problem.

The last part of the thesis presented an example to show the proper draft integrity constraints. ER diagram is created there as well as description of entities and attributes. Moreover, the draft integrity constraints in the chosen example is shown and an implementation with using precepts of SQL language. there is also explains how important is the draft of integrity constraints to worked the database correctly and prevent errors.

Keywords: relational database technology, data integrity, SQL, triggers, design

Obsah

1. Úvod

V mnoha oblastech lidské činnosti je potřeba shromažďovat a evidovat informace a dále je využívat při následných činnostech. Informace jsou pro lidi a hlavně podnikatelské subjekty a firmy jedním z nejcennějších artiklů, které jim slouží k optimalizaci svých procesů a dosahování větších zisků. Již od pradávna platilo, že kdo byl lépe informován o své konkurenci, byl vždy o velký krok před ní. Proto je potřeba uchovávat informace a s rostoucí velikostí a šíří spektra působnosti daného subjektu narůstá počet informací, které je třeba uchovávat pro další využití.

V nedávné době k uchovávání dat sloužili primárně kartotékové systémy, kde se všechny záznamy uchovávaly v papírové podobě. V dnešní době je již většina kartotékových systémů nahrazena databázemi a nejčastěji právě relačními databázemi. Pro komunikaci s relačními databázemi slouží jazyk SQL, který je standardizovaný organizací ANSI a díky tomu je vysoce přenosný mezi různými typy relačních databázi. Mezi nejvyužívanější systémy patří Oracle, MySQL, MS Access a Microsoft SQL server.

2. Cíle

Tato bakalářská práce si klade za cíl v teoretické části vymezit důležité pojmy kolem databází a jazyka SQL a především datové integrity v rámci tohoto jazyka.

Dalším cílem je výběr současného databázového řešení z běžného života, jeho analýza, zhodnocení stavu, návrh funkčního řešení včetně příkazů k vytvoření správných integritních omezení, které by eliminovaly nedostatky současného řešení a finální zhodnocení celé problematiky.

Posledním cílem práce je návrh vlastního řešení se zaměřením na datovou integritu, kde bude autor demonstrovat správný návrh od počátku, objasní veškeré kroky při návrhu a předvede znalosti především z oblasti datové integrity. Také zde uvede příkazy nutné pro vytvoření integritních omezení, které jeho návrh obsahuje.

3. Metodika

Tato práce je vytvořena na základě studia odborné literatury, zabývající se relačně databázovými systémy, především datovou integritou, a jazykem SQL. Získané informace jsou analyzovány, utříděny a sepsány dle cílů této práce. Práce vychází z principu technologie relačně databázového zpracování a především z teorie o integritních omezeních, které jsou náplní této bakalářské práce. Realizace SQL příkazů demonstrovaných v této práci je na bázi databázového systému Oracle, který byl vyučován na univerzitě, pod jejíž záštitou práce vznikla. Zhodnocení současného řešení je pouze pro nástin dnešní tvorby relačních databází. Návrh vlastního řešení není zpracováván na základě požadavku další strany a je vytvořen čistě pro účely demonstrace nabytých vědomostí a příkazů spojených s integritními omezeními.

4. Teoretická východiska práce

4.1 Databáze

Pojem databáze je v dnešní době velice běžným výrazem, neboť dnes ve většině oblastí lidské činnosti je zapotřebí uchovávat a zpracovávat informace, ať už v menším či větším měřítku. Skoro každá organizace potřebuje uchovávat informace o svých činnostech uvnitř i navenek, aby mohla do budoucna z těchto informací provádět dedukce a směřovat ke svému záměru. Dá se proto hovořit o tom, že celá současná společnost je postavena na databázích. Informace se pak uchovávají nejčastěji v počítačích, ale stále zde ještě přebývají z minulých dob kartotékové systémy, které jsou předchůdci moderních počítačových databází.

Václav Vostrovský definuje databázi následovně: *„Báze dat (databáze) představuje jakousi množinu dat vztahující se k určité problematice, přičemž ve většině případů jde o množinu souborů logicky spolu souvisejících. Výrazným rysem počítačových databází je jejich strukturovanost a uspořádanost. Proto je databáze vlastně jakýsi uspořádaný seznam dat (informací) sloužící k co nejjednoduššímu a nejrychlejšímu vyhledávání informací podle určitého klíče.“*[6]

Na doplnění lze použít definici databáze od Michaela J. Hernandeze: *„Databáze je soubor dat používaný k modelování některých typů organizačních struktur nebo organizačních procesů. Vůbec nezáleží na tom, jestli ke shromažďování a ukládání dat používáte papír, nebo počítačový program. Dokud shromažďujete data nějakým organizovaným způsobem, máte databázi.“*[1]

4.2 Relační databázový model

Je nástupcem hierarchického a síťového databázového modelu, které na konci 60. let přestaly splňovat požadavky kladené na kvalitní databáze. Zakladatelem relačního modelu byl Dr. Edgar F. Codd, který byl výzkumným pracovníkem u IBM. Na konci 60. let se zabýval novými metodami pro zpracování velkého množství dat, a jelikož byl původně matematik, snažil se vyřešit celou řadu existujících problémů, jako např. redundance dat, nízká integrita dat a přílišná závislost databázových struktur na jejich fyzické realizaci, pomocí matematických technik a struktur.[1,6,9]

„Dr. Codd svůj relační model formálně představil v červnu 1970 v mezní práci nazvané „Relační model dat pro velké sdílené databanky“ (A Relational Model of Data for Large Shared Databanks). Svůj model založil na dvou matematických disciplínách - teorii množin a predikátové logice prvního řádu. Dokonce i jméno modelu je odvozeno z pojmu „relace“, který je součástí teorie množin.“[1]

„Relační databáze ukládá data ve vztazích, které vidí uživatel jako tabulky. Každý vztah je složen z uspořádaných n-tic, neboli záznamů, a atributů neboli polí. Skutečné uspořádání záznamů, nebo polí v databázi je zcela nepodstatné a každý záznam v tabulce je identifikován polem, které obsahuje unikátní hodnotu.“[1]

Aby bylo možné označit tabulku jako relační, musí splňovat následující podmínky:

- Všechny hodnoty v tabulce musí být elementární (dále nedělitelné na další údaje)
- V tabulce je 1 až n sloupců, jejichž pozice je nevýznamná (pořadí sloupců lze libovolně měnit)
- V tabulce je 0 až m řádků, jejichž pozice je nevýznamná (pořadí řádků lze libovolně měnit)
- Sloupec musí být homogenní (obor hodnot tohoto sloupce je stejný)
- Každý sloupec musí být jednoznačně pojmenován (toto jméno je tzv. atribut)
- Každý řádek tabulky musí být jednoznačně rozlišitelný (v tabulce nesmí existovat dva stejné řádky shodující se ve všech sloupcích)[1]

Více tabulek mezi sebou lze propojovat pomocí operací sjednocení, průnik a kartézský součin.[6]

4.3 Výhody relační databáze

Relační databáze má oproti předchozím modelům několik výhod, např.:

- *„Zabudovaná víceúrovňová integrita. Integrita dat je zabudována přímo do modelu na úrovni položek, aby se zajistila přesnost dat. Na úrovni tabulek zajišťuje, že záznamy nejsou duplicitní a detekuje chybějící hodnoty primárních klíčů. Na úrovni vztahů zajišťuje, že vztah mezi dvojicí tabulek je platný.*
- *Logická a fyzická nezávislost dat na databázové aplikaci. Ani změny uživatele v logickém návrhu, ani změny poskytovatele databázového softwaru ve fyzické*

implementaci databáze neovlivní aplikaci postavenou pro původní logický návrh a databázový software.

- *Garantovaná konzistence a přesnost dat. Data jsou konzistentní a přesná díky různým úrovním integrity, které se dají v databázi vynutit.*
- *Snadné získávání dat. Na základě uživatelova příkazu mohou být data získána buď z určité tabulky, nebo z libovolného počtu tabulek, které jsou ve vztahu. To uživateli umožňuje zobrazovat informace téměř neomezeným počtem způsobů.“*[1]

4.4 Systémy pro správu relační databáze

RDBMS (relational database management system, systém pro správu relační databáze) je program, používající se k vytvoření, udržování, modifikování a správě relační databáze. Kvalita tohoto systému je přímo úměrná tomu, jak podporuje relační databázový model.[9]

V počátcích byly RDBMS používány na sálových počítačích. Dvěma nejobvyklejšími na počátku 70. let byly System R vyvinutý IBM a Interactive Graphics Retrieval System (INGRES) vyvinutý na Kalifornské univerzitě v Berkeley. Oba programy značně přispěly k uznání relačního modelu.[9]

V 80. letech vedl zvýšený tlak podniků, přesouvajících své databáze z hierarchických a síťových na relační, k tvorbě nových a kvalitnějších komerčních RDBMS pro sálové počítače, jako např. Oracle od Oracle Corporation nebo DB2 od IBM.[9]

Když se v polovině 80. let zrodil osobní počítač, přišly i RDBMS určené pro osobní počítače. V počátcích to byly jednoduché systémy pro správu databáze založené pouze na souborech, jako např. dBase od Ashton-Tate a FoxPro od Fox Software. Opravdové RDBMS pro osobní počítače se začaly objevovat až s uvedením R:BASE (původně od Micronim) a paradox (původně od Ansa Software). Tyto dva produkty pomohly rozšířit myšlenku a možnost správy databází ze sálových počítačů na osobní počítače.[9]

Během 80. a 90. let, kdy stále více uživatelů začalo používat databáze, se stala zřejmou potřeba sdílet data. Dobrým nápadem se zdála myšlenka centrálně uložené databáze, která může být zpřístupněna více uživatelům. Producenti databázových systémů

na tuto myšlenku odpověděli vývojem RDBMS klient/server. Data jsou uložena na počítači sloužícím jako databázový server a uživatelé pracují na vlastních počítačích, neboli databázových klientech, kde zpracovávají data pomocí aplikací zde spuštěných. Vývojář databáze používá RDBMS k vytvoření a správě databáze a vytvoření provázané aplikace pro koncového uživatele. RDBMS klient/server se využívají pro správu velkých množství sdílených dat. Zástupci jsou např. Microsoft SQL Server 2000 od Microsoft Corporation a Oracle9i Application Server od Oracle Corporation.[9]

4.5 Pojmy vztahující se ke struktuře

4.5.1 Tabulka

Hlavními strukturami v databázi jsou právě tabulky. Každá tabulka vždy reprezentuje jednu entitu. V tabulce nemá vůbec žádný význam logický sled záznamů a polí. Vždy musí obsahovat alespoň jedno pole známé jako primární klíč, který jednoznačně identifikuje všechny její záznamy.[1,6]

Entita, kterou reprezentuje daná tabulka, může být buď objekt, nebo událost. Pokud je entita objektem, pak tabulka reprezentuje hmatatelnou věc, například osobu či místo. Každý objekt, bez ohledu na svůj typ, má nějaké vlastnosti, a ty lze uložit jako data. Tyto data lze pak zpracovávat téměř jakýmkoli možným způsobem.[1]

Pokud tabulka reprezentuje něco, co se událo v nějakém čase a co má jisté vlastnosti, které chceme zaznamenat, je entitou tabulky událost. Příkladem je soudní slyšení či výsledky laboratorních testů. Vlastnosti takto zaznamenané mohou být uloženy jako data, a poté zpracovány jako informace přesně stejným způsobem jako tabulka reprezentující určitý objekt.[1]

Zákazníci

Zákazník ID	Křestní jméno klienta	Příjmení klienta	Město klienta	Další pole
9001	Stewart	Jameson	Seattle	-----
9002	Shannon	McLain	Poulsbo	-----
9003	Estela	Pundt	Tacoma	-----
9004	Timothy	Ennis	Seattle	-----
9005	Marvin	Russo	Bellingham	-----
9006	Kendra	Bonnicksen	Tacoma	-----

Záznamy

Pole

Obrázek 1 - Typická tabulková struktura [1]

Tabulka, která ukládá data určená k dodávání informací, se nazývá datová tabulka a je nejčastějším typem tabulky v relační databázi. V tomto typu tabulky jsou data dynamická, protože s nimi manipulujeme, měníme je nebo mažeme, a zpracováváme je na informace.[1]

Oproti tomu validační tabulka ukládá data sloužící k implementaci integrity dat. Obvykle reprezentuje entity jako například kódy produktů či jména měst. V tomto typu tabulek jsou data statická, protože se mění jen zřídka. Nejčastěji se používají ke kontrolování platnosti hodnot, které se zadávají do datových tabulek.[1]

4.5.2 Pole

Nejmenší strukturou v databázi je pole neboli atribut, které reprezentuje nějakou vlastnost entity odpovídající tabulce, do které patří. Struktury, které ve skutečnosti ukládají data, jsou pole. Získávání a prezentování dat z nich na informace lze téměř v jakékoli konfiguraci, kterou si lze představit. Kvalita získávaných a prezentovaných informací z uložených dat, je v přímé úměře k času stráveném k zajištění strukturální a datové integrity polí.[1,6]

Vícesložkové pole

Vypočítané pole

Vícehodnotové pole

Zákazníci

Zákazník ID	Křestní jm. klienta	Příjmení klienta	Celé jm. klienta	Adresa	Město, stát, PSČ	Poradce
9001	Stewart	Jameson	Stewart Jameson	-----	Seattle, WA, 98125	John, Sandi
9002	Shannon	McLain	Shannon McLain	-----	Pouleno, WA, 98370	Frits
9003	Estela	Pundt	Estela Pundt	-----	Tacoma, WA, 98005	John
9004	Timothy	Ennis	Timothy Ennis	-----	Seattle, WA, 98115	Frits, Sandi
9005	Marvin	Russo	Marvin Russo	-----	Bellingham, WA, 98225	Frits, John
9006	Kendra	Bonnicksen	Kendra Bonncksen	-----	Olympia, WA, 98504	Sandi

Obrázek 2 - Tabulka obsahující korektní, vypočítané, vícesložkové a vícehodnotové pole [1]

Ve správně navržené databázi každé pole obsahuje právě jednu hodnotu a jeho jméno přesně identifikuje typ údaje v něm uloženém. Zadávání dat do polí je pak velmi intuitivní.[1]

Pokud je databáze špatně navržená, nalezneme v ní tři typy polí:

1. Vícesložkové pole (složené pole) obsahující ve své hodnotě dvě nebo více součástí.
2. Vícehodnotové pole obsahující několik hodnot stejného typu.

3. Vypočítané pole obsahující spojené textové řetězce, nebo výsledky matematických operací.[1]

4.5.3 Záznam

Unikátní instance entity dané tabulky je záznam. Složen je v tabulce z celé řady polí, bez jakéhokoli ohledu na to, zda pole obsahují nějaké hodnoty nebo ne. V databázi je každý záznam identifikován jedinečnou hodnotou v primárním klíči. V pochopení vztahů mezi tabulkami hrají klíčovou roli právě záznamy, jelikož je důležité vědět, v jakém vztahu je záznam v jedné tabulce k záznamu v jiné tabulce.[1,6]

4.5.4 Klíče

Speciálními poli hrající v tabulkách významnou roli jsou klíče. Typ klíče určuje jeho význam v tabulce. Jsou dva typy klíčů, primární a cizí.[1,6]

Jednoznačný identifikátor každého záznamu v tabulce je primární klíč, který je buď jedno pole, nebo skupina polí. Pokud je složen ze dvou nebo více polí, pak se označuje jako složený primární klíč. Je nejdůležitějším klíčem v tabulce. V celé databázi identifikuje daný záznam právě hodnota primárního klíče. V rámci celé databáze identifikuje danou tabulku pole primárního klíče. Integritu na úrovni tabulky vynucuje primární klíč a pomáhá s ostatními tabulkami v databázi zřizovat vztahy.[1,6]

Primární klíč

Agenti

Agent ID	Křestní jméno agenta	Příjmení agenta	Datum přijetí	Agentův telefon
100	Stella	Rosales	16.5.1995	553-3992
101	Steve	Pundt	15.10.1995	790-3992
102	Randi	Nathanson	1.3.1996	551-4993

Primární klíč Cizí klíč

Baviči

Bavič ID	Agent ID	Jméno baviče	Telefon baviče	Další pole
9001	100	Jazz Time	555-9928	-----
9002	101	Trio Mika Hernandeze	959-8837	-----
9003	100	Páni venkova	709-3542	-----

Obrázek 3 - Příklad polí primárních a cizích klíčů [1]

Cizí klíč je kopie primárního klíče v jiné tabulce, než ze které pochází a se kterou je v nějakém vztahu. Cizím klíčem v druhé tabulce se označuje proto, že daná tabulka již svůj primární klíč má a že primární klíč první tabulky je z pohledu druhé tabulky cizí.[1,6]

Cizí klíč pomáhá se zřizováním vztahů mezi dvojicemi tabulek, ale také implementovat a zajišťovat integritu dat na úrovni vztahů. Což znamená, že záznamy v obou tabulkách jsou vždy ve správném vztahu, jelikož hodnota cizího klíče musí odpovídat hodnotě primárního klíče, na který odkazuje. Předcházení obávaným tzv. sirotčím záznamům napomáhá právě integrita na úrovni vztahů, například koncert nemá žádného účinkujícího. Pokud na koncertě nikdo neúčinkuje, zajisté nedorazí žádní diváci a zisk z koncertu nebo kulturní zážitek nebude žádný.[1,6]

4.6 Vztahy

Pokud lze nějakým způsobem propojit záznamy z první tabulky se záznamy z druhé tabulky, pak mezi nimi existuje vztah. Vztah lze zřídit buď to prostřednictvím primárních a cizích klíčů nebo přes další tabulku, která se nazývá vazební tabulkou nebo také asociativní tabulkou. Způsob, jakým je zřizován vztah, je dán typem vztahu, který mezi tabulkami existuje. Důležitou součástí relační databáze je vztah, jelikož umožňuje vícetabulkové pohledy a je klíčový pro integritu dat, z důvodů redukce nadbytečných dat a eliminace duplicitních dat.[1,6]

Každý vztah lze charakterizovat třemi způsoby:

1. Typem vztahu, který existuje mezi tabulkami
2. Způsob, jakým se tabulky účastní vztahu
3. Stupeň, jakým se účastní[1]

4.6.1 Typy vztahů

Jsou tři typy vztahů (též označované jako kardinality):

1. 1:1
2. 1:N
3. M:N[1,6]

4.6.1.1 Vztah 1:1

„Dvojice tabulek je ve vztahu 1:1, pokud jeden záznam v první tabulce je ve vztahu pouze k jednomu záznamu ve druhé tabulce a jeden záznam ve druhé tabulce je ve vztahu k pouze jednomu záznamu v první tabulce. V tomto typu vztahu slouží jedna tabulka jako rodičovská a druhá jako tabulka potomka. Uvedený vztah zřídíte tak, že vezmete kopii primárního klíče rodičovské tabulky a začleníte jí do struktury tabulky potomka, kde se stane cizím klíčem. Je to speciální případ vztahu, protože pouze v tomto případě mohou obě tabulky sdílet primární klíč.“[1]

Zaměstnanci

ZaměstnanecID	Křestní jméno zaměstnance	Příjmení zaměstnance	Telefon domů	Další pole
100	Zachary	Erlich	553-3992	-----
101	Susan	McLain	790-3992	-----
102	Joe	Rosales	551-4993	-----

Náhrady

ZaměstnanecID	Plat za hodinu	Procento odměny	Další pole
100	25,00	5,0%	-----
101	19,75	3,5%	-----
102	22,50	5,0%	-----

Obrázek 4 - Příklad vztahu 1:1 [1]

4.6.1.2 Vztah 1:N

„Vztah 1:N existuje mezi dvěma tabulkami, pokud může být jeden záznam v první tabulce ve vztahu k mnoha záznamům v druhé tabulce, ale záznam v druhé tabulce může být ve vztahu pouze k jednomu záznamu v první tabulce. (Model rodič/potomek se dá použít i v tomto případě. Tabulka na straně 1 je rodič a tabulka na straně N je potomek.) Vztah 1:N se zřídí zkopírováním primárního klíče tabulky rodiče a začleněním do tabulky potomka, kde se stane cizím klíčem.“[1]

Tento vztah je obecně nejobvyklejším typem vztahu, který existuje v databázi mezi dvěma tabulkami. Z pohledu datové integrity je klíčový, jelikož napomáhá k eliminaci duplicitních dat a minimalizaci nadbytečných dat.[1]

Agenti

Agent ID	Křestní jméno agenta	Příjmení agenta	Datum přijetí	Agentův telefon
100	Stella	Rosales	16.5.1995	553-3992
101	Steve	Pundt	15.10.1995	790-3992
102	Randi	Nathanson	1.3.1996	551-4993

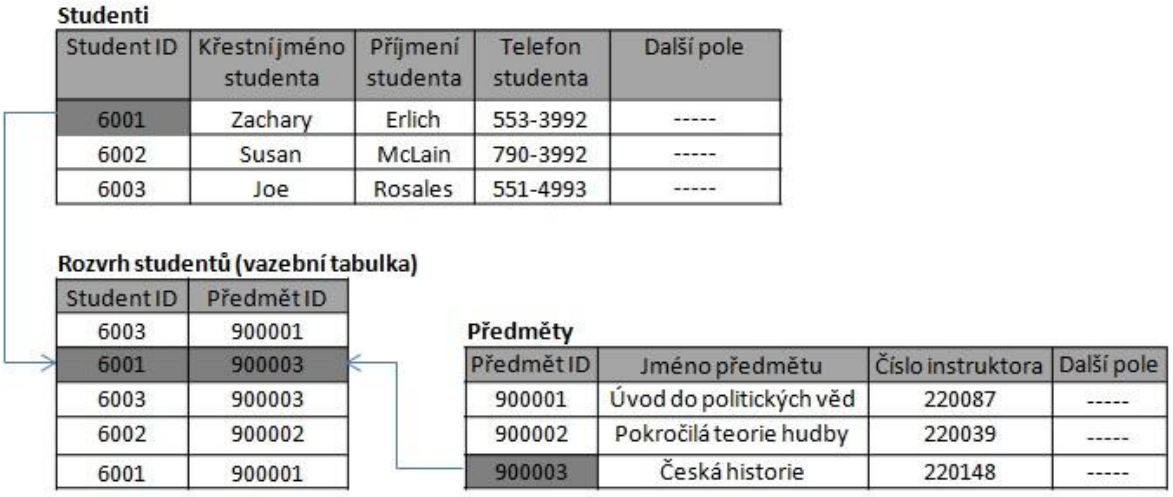
Baviči

Bavič ID	Agent ID	Jméno baviče	Telefon baviče	Další pole
9001	101	Jazz Time	555-9928	-----
9002	100	Trio Mika Hernandeze	959-8837	-----
9003	100	Páni venkova	709-3542	-----

Obrázek 5 - Příklad vztahu 1:N [1]

4.6.1.3 Vztah M:N

„Dvojice tabulek má k sobě vztah M:N, v případě, že jeden záznam v první tabulce může být ve vztahu k mnoha záznamům ve druhé tabulce a záznam ve druhé tabulce může být ve vztahu k mnoha záznamům v první tabulce. Tento vztah se zřizuje pomocí vazební tabulky. Vazební tabulka usnadňuje navázání záznamů z jedné tabulky na záznamy z druhé tabulky a zajišťuje, že nejsou žádné problémy s přidáváním, mazáním nebo upravováním souvisejících dat. Vazební tabulka se vytvoří pomocí spojení kopií primárních klíčů obou původních tabulek do nové tabulky. Tato pole vlastně hrají dvě úlohy. Spolu tvoří složený primární klíč vazební tabulky. Každé odděleně potom tvoří cizí klíč.“[1]



Obrázek 6 - Řešení vztahu M:N pomocí vazební tabulky [1]

4.7 Integrita dat

V databázi integrita dat označuje platnost, konzistentnost a přesnost dat. Důležitým faktem je, že stupeň přesnosti získávaných informací z databáze je v přímé úměře ke stupni integrity uložených dat v databázi. Jedním z nejdůležitějších aspektů procesu logického návrhu databáze je právě integrita dat, která se nesmí podceňovat, přehlížet nebo dokonce částečně opomíjet. Tento přístup by vedl k nebezpečnému množství chyb, které by šlo velmi těžko nalézt a odstranit. Výsledkem tohoto počínání bude rozhodování na základě informací, které mohou být v lepším případě nepřesné, v horším případě neplatné.[1,6]

Jsou čtyři typy integrity, které se zavádějí v průběhu procesu návrhu databáze. Tři první jsou označeny podle oblasti, kde se vyskytují, a jsou založeny na různých aspektech struktury databáze. Čtvrtý typ je založen na způsobu, jakým organizace chápe a užívá svá data. Zde je stručný popis všech typů:

1. Integrita na úrovni tabulky (integrita entity) zajišťuje, že neexistují záznamy duplicitní, a že pole, které každý záznam v tabulce identifikuje, nikdy neobsahuje null a je jedinečné.
2. Integrita na úrovni pole (doménová integrita) zajišťuje, že struktura v každém poli je spolehlivá, že hodnoty uložené v každém poli jsou platné, přesné a konzistentní a že pole stejného typu jsou definovány v celé databázi konzistentně.

3. Integrita na úrovni vztahů (referenční integrita) zajišťuje, že mezi dvěma tabulkami je spolehlivý vztah a že v obou tabulkách jsou záznamy synchronizovány kdykoli jsou v kterékoli z tabulek data zadávána, mazána nebo opravována.
4. Business pravidla jsou založené na způsobu, jakým organizace získává a používá data a zavádí omezení na jisté aspekty databáze. Tato omezení mohou ovlivnit při návrhu databáze některé aspekty, jako např. typ účasti a stupeň účasti každé tabulky ve vztahu, rozsah a typ hodnot uložených v poli nebo typ synchronizace používaný pro integritu na úrovni vztahů v některých vztazích. Jelikož ovlivňují integritu, musejí se brát v průběhu procesu návrhu v úvahu kdykoli ovlivňujeme integritu dat.[1]

4.7.1 Omezení sloupce

„Omezení sloupce určitým způsobem omezují (limitují) hodnoty, které lze do sloupce tabulky umístit.“[2] Tato omezení mohou nabývat nějaké z následujících forem:

- *„Klauzule DEFAULT. Tento výraz se aplikuje na hodnotu sloupce při vložení nového řádku, který hodnotu daného sloupce neobsahuje. Může se jednat o libovolný platný výraz jazyka SQL, např. konstantu, funkci SQL nebo jiný výraz, který po vyhodnocení modulem SQL v SŘBD poskytne správnou datovou hodnotu příslušného sloupce.“*[2]
- Omezení NULL nebo NOT NULL. Klíčové slovo NULL dovoluje, aby sloupec obsahoval hodnoty null, zatímco NOT NULL tyto hodnoty zakazuje. V SŘBD Oracle platí, že pokud klauzule není uvedena, databáze bude předpokládat hodnotu NULL.[2]
- *„Omezení CHECK. Pomocí omezení CHECK lze vynutit libovolné pravidlo, které se vztahuje na jednotlivý sloupec tabulky. Podmínka uvedená v omezení musí platit při každé změně dat sloupce tabulky. V opačném případě SŘBD změnu odmítne a zobrazí chybovou zprávu. Důležité omezení spočívá v tom, že podmínka omezení sloupce nesmí odkazovat na žádný jiný sloupec.“*[2]
- Omezení UNIQUE. Toto omezení zaručuje, že hodnoty příslušného sloupce v tabulce budou jedinečné, obvykle pomocí indexu, který automaticky vytváří SŘBD.[2]

- *„Omezení PRIMARY KEY. Omezení primárního klíče určuje, že sloupec slouží jako primární klíč tabulky. Toto omezení vyžaduje, aby sloupec neobsahoval žádné hodnoty null a jeho hodnoty v rámci tabulky byly jedinečné. Stejně jako u omezení UNIQUE SŘBD automaticky vytvoří index, který zajišťuje kontrolu jedinečnosti datových hodnot ve sloupci.“*[2]
- *„Referenční omezení (FOREIGN KEY). Referenční omezení sloupce (někdy se označuje jako omezení cizího klíče) definuje vztah mezi cizím klíčem a primárním klíčem. Díky tomu může SŘBD zaručit, že se hodnota cizího klíče (různá od hodnoty null) vždy odkazuje na existující hodnotu primárního klíče.“*[2]

4.7.2 Omezení tabulky

„Všechna omezení sloupce lze alternativně zapsat formou jako omezení tabulky. Klauzule, která definuje omezení, je v tomto případě uvedena za všemi definicemi sloupců v příkazu CREATE TABLE, nikoliv uprostřed definice sloupce. Hlavní výhoda omezení tabulky před omezením sloupce spočívá v tom, že omezení tabulky se může vztahovat na více sloupců.“[2]

4.8 Jazyk SQL

„Historie jazyka SQL spadá do 70. a 80. let. První standard byl přijat v roce 1986 (označován jako SQL86*). Časem se však projevily některé nedostatky. Opravená verze je z roku 1992 a je označována jako* SQL92*. Ten je v oblasti relačních databází standardem dodnes. Zkratka SQL značí Structured Query Language. Jazyk v sobě zahrnuje nástroje pro tvorbu databází (tabulek) a dále nástroje na manipulaci s daty (vkládání dat, aktualizace, mazání a vyhledávání informací).“*[3]

SQL spadá pod tzv. deklarativní programovací jazyky, to znamená, že kód jazyka SQL se nepíše v žádném samostatném programu, například jako tomu je u jazyka C nebo Pascal, ale je vkládán do jiného programovacího jazyka, který je již procedurální. Se samotným jazykem SQL je možné pracovat pouze v případě, kdy se terminálem připojíme na SQL server a na příkazový řádek přímo zadáváme příkazy jazyka SQL.[3]

SQL se skládá z několika částí. Části určené pro administrátory a návrháře databázových systémů, a dále pak části určené pro koncové uživatele a programátory. První částí je jazyk DDL (Data Definition Language), který slouží k vytváření

databázových schémat a katalogů. Druhou částí je jazyk SDL (Storage Definition Language), který definuje způsob ukládání tabulek. Třetí částí je jazyk VDL (View Definition Language) pro návrháře a správce, díky kterému lze přidělovat práva a role a vytvářet pohledy (pohled si lze představit jako virtuální tabulku složenou z jiných tabulek). Čtvrtou částí je jazyk DML (Data Manipulation Language) pro koncové uživatele a programátory databázových aplikací.[3,6]

4.9 Triggery

Trigger je nástroj, zajišťující automatické provedení programu v jazyce SQL při vložení, změně nebo zrušení záznamu v určité tabulce. Může například zajistit, že se před vložením nového záznamu do tabulky provede kontrola vkládaného obsahu, po smazání záznamu se provedou další akce nebo při změně hodnoty v určitém sloupci záznamu se porovná stará a nová hodnota a na základě toho se pak provedou další akce.[4]

Zjednodušují tvorbu aplikací, jelikož část práce databázové aplikace přenášejí na server. Pro informační systém lze triggery centralizovaně definovat platná pravidla.[4]

„Trigger je pojmenovaným objektem patřícím do databázové aplikace. Má tyto vlastnosti:

- *Je svázán s určitou tabulkou a reaguje na změny v této tabulce*
- *Reaguje na právě jednu z SQL akcí INSERT, DELETE nebo UPDATE, triggery reagující na UPDATE mohou navíc specifikovat množinu sloupců, jejichž současná změna (všech vyjmenovaných) je spouští*
- *Provádí se buď před (BEFORE) provedením výše specifikované akce, nebo po ní (AFTER)*
- *Trigger nelze vyvolat editací hodnot multiatributu*
- *Pokud akce ovlivňuje více než jeden záznam, pak se může spouštět buď pro každý ovlivněný (tj. vložený, zrušený nebo změněný) záznam zvlášť, nebo pro všechny záznamy najednou*
- *Může obsahovat podmínku, která se vyhodnotí před provedením triggeru a pokud není splněna, trigger nebude spuštěn*
- *Specifikuje akci (zapsanou v jazyce SQL), která bude automaticky provedena, jakmile jsou splněny podmínky pro spuštění triggeru“*[4]

V momentě provádění konkrétní akce lze spustit více než jeden trigger na základě splnění podmínek pro jejich spuštění. V případě více triggerů přiřazených ke konkrétní akci, se spouštějí všechny po sobě, nikoliv však více v jeden moment. Při provádění triggerů lze měnit obsah databáze, což znamená, že jeden trigger může spouštět další trigger, který je do něj vnořený.[4]

„Při provádění triggerů se nekontrolují práva. Aby díky tomu nemohlo dojít k neoprávněnému zápisu, může triggery vytvářet pouze správce databáze resp. správce aplikace.“[4]

5. Současná úroveň řešené problematiky

5.1 Registr vozidel

Jako příklad z praxe, na kterém bude demonstrováno dnešní řešení databázových aplikací, byl vybrán centrální registr vozidel, který je znám svými přetrvávajícími problémy a zajisté špatným a ledabylým zpracováním. Na firmu ATS-Telcom, dodavatele centrálního registru vozidel, se snášela a dále snáší ze všech stran tvrdá kritika kvůli nefunkčnosti a chybovosti celého systému. Jednou z příčin je také to, že firma ATS-Telcom doposud neměla žádné zkušenosti s tvorbou podobných systémů.

Článek v dopravních novinách ze dne 26.7.2012 uvádí:

„V novém registru vozidel je téměř půl milionu aut se špatně uvedenými údaji o majiteli. V tiskové zprávě to dnes uvedlo ministerstvo dopravy. Celkově je v registru více než milion chyb. „Věděli jsme, že podobná data se v systému nachází, ale nečekali jsme, že jich bude přes milion," uvedl ministr dopravy Pavel Dobeš. Tyto záznamy bude prý nutné ručně vyčistit a utřídit, což udělá dodavatel, společnost ATS-Telcom. Té tato povinnost společně s povinností údržby vyplývá ze smlouvy, dodal.

Podle ministerstva je v registru 497 460 záznamů s chybně vloženou identifikací majitele vozidla. Jde například o to, že se v systému nezobrazuje aktuální majitel vozidla, ale předchozí majitel, řekl Zdeněk Neusar z tiskového oddělení ministerstva dopravy. U dalších 414 888 záznamů není možné majitele identifikovat při srovnání s dalšími databázemi.

Registrační značka nesouhlasí v téměř 21 000 případech a v 7000 případech se v databázi objevily duplicitní VIN kódy. To může znamenat, že na jeden VIN kód mohlo být vydáno několik technických průkazů na několik různých jmen, uvedl Dobeš."[5]

Zde je patrné, že v případě 497 460 záznamů s chybně vloženou identifikací majitele vozidla, nebyla správně nastavena referenční integrita, a proto zde dochází k problémům, že dané vozidlo vlastní nikoliv současný majitel, ale majitel předešlý. Zde může nastat problém v případě, že bude nutno stíhat majitele vozidla za nějaký

přestupek, avšak místo pravého majitele se může do potíží dostat bývalý majitel, který s daným vozidlem nemá již nic společného.

V případě 414 888 záznamů s nemožností identifikace majitele vozu při porovnávání s dalšími databázemi je zde problém, jak s referenční integritou, tak i s doménovou integritou. Nejspíše se díky tomu data při převodu ze starých databází do nového registru pomíchala a tak vznikli neexistující majitelé, které není možné dohledat, což znovu zapříčiňuje velké obtíže v případě jakýchkoliv problémů s daným vozidlem. Ale samozřejmě také problémy se správností dat, doménovou integritou, které tomu předcházejí.

U registračních značek je zde opět problém v referenční integritě, jako v předešlých případech. Duplicitní VIN kódy jsou porušením entitní integrity. Rozhodně není možné, aby po světě jezdili dvě vozidla se stejným VIN kódem, jelikož ten je u automobilů něco jako primární klíč, a tak by k němu mělo být přistupováno i v databázi, měl by být jedinečný.

5.2 Zjednodušené ukázkové řešení registru vozidel

Zde bude demonstrováno, jak by měl být registr vozidel navržen, aby se předešlo chybám, které registr obsahuje a obsahoval. Vše bude vytvořeno na základě znalostí o dané problematice z pohledu člověka vlastnícího řidičský průkaz a vozidlo, a znalostí tvorby korektní databáze, jelikož přístup k nahlédnutí do daného registru nebyl možný.

5.2.1 Popis entit a jejich atributů

5.2.1.1 Registrační značka

Tabulka registracni_znacka obsahuje atribut reg_zn, který slouží jako primární klíč. Dále zde je atribut dat_vydani datového typu date, definován jako null a atribut kraj datového typu char, definován jako not null. Slouží k evidenci data vydání registrační značky a kraje, dané registrační značky.

registracni_znacka			
reg_zn	char	pk	not null
dat_vydani	date		null
kraj	char		not null

Tabulka 1 - Registrační značka

5.2.1.2 Majitel

Tabulka majitel obsahuje atribut rodne_cislo, který je jednoznačným identifikátorem každého řádku, a proto je primárním klíčem. Dále pak tabulka obsahuje doplňující atributy o majiteli, kterými jsou jmeno, prijmeni, ulice, mesto a psc. Všechny tyto atributy jsou datového typu char, kromě atributů ulice a město, jelikož některé české ulice a města mají dlouhý název, a proto je varchar vhodnějším řešením a všechny jsou definovány jako not null, jelikož jsou to důležité informace o vlastníkovi vozidla.

majitel			
rodne_cislo	char	pk	not null
jmeno	char		not null
prijmeni	char		not null
ulice	varchar		not null
mesto	varchar		not null
psc	char		not null

Tabulka 2 - Majitel

5.2.1.3 Vozidlo

Tabulka vozidlo obsahuje jako primární klíč atribut vin (vehicle identification number). Atribut reg_zn je cizím klíčem odkazujícím na tabulku registracni_znacka a je definován jako unique. Atribut rodne_cislo je dalším cizím klíčem odkazujícím na tabulku majitel a je definován jako not null. Dalšími atributy jsou znacka, model, rok_vyr a barva. Kromě atributu rok_vyr, který je datového typu number, jsou všechny datového typu char a jsou definovány jako not null, protože blíže specifikují dané vozidlo.

vozidlo			
vin	char	pk	not null
reg_zn	char	fk	unique
rodne_cislo	char	fk	not null
znacka	char		not null
model	char		not null
rok_vyr	number		not null
barva	char		not null

Tabulka 3 - Vozidlo

5.3 Vytvoření tabulek a jejich integritních omezení

Budou zde uvedeny příkazy pro vytvoření tabulek, které jsou uvedeny výše, včetně jejich integritních omezení. Následně bude demonstrováno naplnění tabulek ukázkovými záznamy.

5.3.1 Vytvoření tabulek

```
CREATE TABLE registracni_znacka (
        reg_zn CHAR(7) PRIMARY KEY NOT NULL,
        dat_vydani  DATE NULL,
        kraj CHAR(15) NOT NULL);
CREATE TABLE majitel (
        rodne_cislo CHAR(11) PRIMARY KEY NOT NULL,
        jmeno  CHAR(15) NOT NULL,
        prijmeni  CHAR(15) NOT NULL,
        ulice VARCHAR NOT NULL,
        mesto VARCHAR NOT NULL,
        psc CHAR(15) NOT NULL);
CREATE TABLE vozidlo (
        vin CHAR(17) PRIMARY KEY NOT NULL,
        reg_zn CHAR(7) UNIQUE,
        rodne_cislo CHAR(11) NOT NULL,
        znacka CHAR(10) NOT NULL,
        model CHAR(10) NOT NULL,
        rok_vyr NUMBER(4) NOT NULL,
        barva CHAR(10) NOT NULL,
        FOREIGN KEY (reg_zn) REFERENCES registracni_znacka(reg_zn)
        FOREIGN KEY (rodne_cislo) REFERENCES majitel(rodne_cislo));
```

5.3.2 Naplnění tabulek

INSERT INTO registracni_znacka VALUES ('5A39185', '1998-09-15', 'Praha');

INSERT INTO registracni_znacka VALUES ('3A21256', '2010-12-19', 'Praha');

INSERT INTO registracni_znacka VALUES ('1P34598', '2011-01-21', 'Plzeňský');

INSERT INTO majitel VALUES ('900304/0165', 'Jan', 'Novák', 'Na Louce 3', 'Praha 4', '14800');

INSERT INTO majitel VALUES ('751011/0206', 'Jana', 'Svobodová', U pivovaru 21', 'Plzeň', '30100');

INSERT INTO vozidlo VALUES (' TMB123456A0123456', '1P34598', '751011/0206', 'Škoda', 'Octavia', 2011, 'modrá', 'osobní');

INSERT INTO vozidlo VALUES (' WVWZZZ3BZWE689725', '5A39185', '900304/0165', 'VW', 'Passat', 1998, 'černá', 'osobní');

INSERT INTO vozidlo VALUES (' WAUZZZ8K6AA103083', '3A21256', '900304/0165', 'Audi', 'A4', 2010, 'červená', 'osobní');

5.3.3 Zobrazení naplněných tabulek

vozidlo

vin	reg_zn	rodne_cislo	znacka	model	rok_vyr	barva
TMB123456A0123456	1P34598	751011/0206	Škoda	Octavia	2011	modrá
WVWZZZ3BZWE689725	5A39185	900304/0165	VW	Passat	1998	černá
WAUZZZ8K6AA103083	3A21256	900304/0165	Audi	A4	2010	červená

Tabulka 4 - Záznamy tabulky vozidlo

majitel

rodne_cislo	jmeno	prijmeni	ulice	město	psc
900304/0165	Jan	Novák	Na Louce 3	Praha 4	14800
751011/0206	Jana	Svobodová	U Pivovaru 21	Plzeň	30100

Tabulka 5 - Záznamy tabulky majitel

registracni_znacka

reg_zn	dat_vydani	kraj
5A39185	15.9.1998	Praha
3A21256	19.12.2010	Praha
1P34598	21.1.2011	Plzeňský

Tabulka 6 - Záznamy tabulky registrační značka

5.3.4 Otestování vkládaných dat

Zde bude demonstrováno několik příkladů na otestování funkčnosti nadefinovaných omezení, které řeší problémy, jenž v registru vozidel nebyly podchyceny.

5.3.4.1 Vícekrát použitá registrační značka

INSERT INTO vozidlo VALUES (' TMB654321A6543210', '1P34598', '751011/0206', 'Škoda', 'Superb', 2013, 'zelená', 'osobní');

Pokud bychom nyní chtěli vložit záznam uvedeným příkazem výše, kde bychom nějakému vozidlu chtěli přiřadit již obsazenou registrační značku, narazili bychom na integritní omezení unique v tabulce vozidlo u atributu reg_zn a záznam by nebyl vložen. Bylo by proto nemožné, aby se v databázi objevili dvě vozidla se stejnou registrační značkou, což je hrubý nedostatek, jelikož v reálném světě toto legálně možné není.

5.3.4.2 Majitel s více vozidly

INSERT INTO vozidlo VALUES (' WVWZZZ3BZWE689725', '5A39185', '900304/0165', 'VW', 'Passat', 1998, 'černá', 'osobní');

INSERT INTO vozidlo VALUES (' WAUZZZ8K6AA103083', '3A21256', '900304/0165', 'Audi', 'A4', 2010, 'červená', 'osobní');

Příkazy výše jsme vložili do databáze dvě vozidla se stejným majitelem. V reálném světě může jedinec vlastnit několik vozidel a proto u atributu rodne_cislo v tabulce vozidlo nebylo uvedeno unique, jako v předchozím případě právě z tohoto důvodu.

5.3.4.3 Duplicitní VIN kódy

Jelikož byl VIN kód zvolen jako primarní klíč tabulky vozidlo, není možné, aby nastala situace, kdy se v databázi objeví dvě vozidla se stejným kódem VIN. Prímární klíč je vždy jednoznačným identifikátorem každého záznamu, a proto tato možnost nemůže nastat a databáze by jakýkoliv pokus o vložení duplicitního VIN kódu neumožnila.

5.4 Závěrečné zhodnocení současné úrovně

Firma ATS-Telcom se obhajovala, že s chybami v databázi nemá nic společného a za nově vzniklé chyby mohou úředníci, kteří s databází nesprávně pracují. Také se odkazovala na nedostatek času pro vytvoření registru vozidel. Pokud pomineme veškeré

nedostatky, které nemají s databází a především datovou integritou a správností dat cokoliv společného, jako například nevhodné a málo výkonné servery, špatnou architekturu, nevhodné uživatelské prostředí a další problémy, kde se nedá o pochybení ze strany zhotovitele nijak pochybovat, chyba i na straně databází bude pocházet od firmy ATS-Telcom. Nejspíše nastal problém v malé analýze skutečných potřeb a procesů nutných pro správu registru vozidel, proto úředníci nemohli vykonávat práci kvalitně, jak byli zvyklí. Na základě této analýzy, byla navrhnuta špatná integritní omezení, jež měla za následek fatální chyby v samotném registru vozidel, příkladem by mohly být duplicitní VIN kódy, nesouhlasící registrační značky, atd.. Dalším velkým problémem byl převod dat do jednotné databáze registru vozidel z jednotlivých registrů po celé republice, kde se pomíchaly veškeré údaje a nebrali se pouze platné (nejaktuálnější) údaje. To mělo za následek například to, že vozidlo najednou vlastnil bývalý majitel nikoliv současný nebo také pomíchání údajů několika majitelů, kteří se stali nedohledatelnými při porovnávání s jinými databázemi.

Zhotovitel registru tedy dle malé vstupní analýzu procesů registru, navrhl špatné integritní omezení anebo je zde možnost, že byla integritní omezení zanedbána z důvodu nekvalifikace pracovníků dané firmy a tento nekvalitní výtvor byl vypuštěn do ostrého provozu, kde neměl šanci správně fungovat a plnit to, co bylo požadováno. Pokud se firma odkazuje na nedostatek času pro vytvoření, neměla danou zakázku od státu přebírat a za těchto podmínek ji zpracovávat. Dle mého názoru firma hrubě pochybila ve všech oblastech, které byly požadovány a zakázku měla přenechat zkušenějšímu subjektu, který by se zajisté podobných chyb vyvaroval, jelikož by měl zkušenosti s obdobným projektem a kvalitnější pracovníky.

6. Vlastní zpracování

6.1 Popis projektu

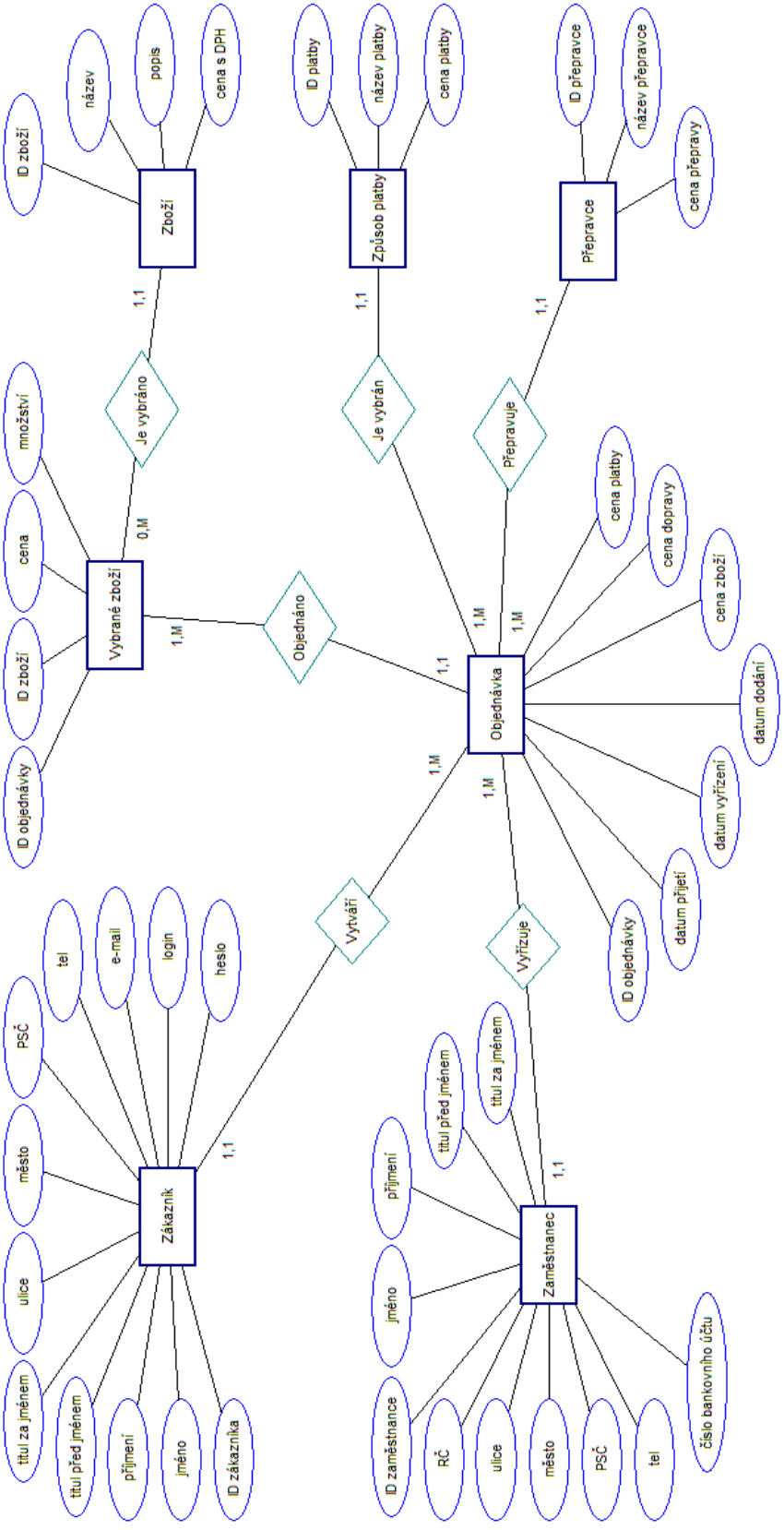
Na předvedení správného návrhu a využití integritních omezení v relačních databázích byl zvolen zjednodušený model internetového obchodu s elektronikou, který by měl být jako ukázka dostačující. Není zbytečně komplikovaný a každý dokáže tento zjednodušený model pochopit, jelikož již v nějakém internetovém obchodě zajisté nakupoval.

Jedná se o internetový obchod, který svým zákazníkům zprostředkovává prodej různých druhů elektroniky (nabízí širokou škálu sortimentu). Tento obchod však nemá svou kamennou pobočku ani výdejní místo a působí pouze na internetu, tudíž všechno jeho zboží musí být k zákazníkům dopravováno pomocí přepravních společností.

Nejdůležitějšími položkami k evidenci jsou zákazníci a jejich objednávky, jelikož je třeba k prodeji tyto informace mít a v případě jakýchkoliv problémů je zpětně dohledat. Dále pak evidovat zboží, které obchod nabízí z důvodu aktuálnosti nabízeného sortimentu. A v poslední řadě mít přehled o svých zaměstnancích, kteří objednávky zákazníků zpracovávají.

Uživateli tohoto systému budou zaměstnanci internetového obchodu, z důvodů vyplývajících z předchozího odstavce (informace o objednávkách, zákaznících, zboží nebo pro případ jakýchkoliv problémů) a majitel internetového obchodu, který potřebuje mít přehled o chodu svého obchodu.

6.2 ER model



Obrázek 7 - ER model

6.3 Popis ER modelu

Entitní množina objednávka je základem modelu. Každá objednávka patří vždy jednomu zákazníkovi, avšak každý zákazník může mít více objednávek. Objednávku vyřizuje zaměstnanec a je zde stejná kardinalita, jako u předchozího vztahu se zákazníkem. Další entitní množinou je zboží. Jelikož jedno zboží může být ve více objednávkách a objednávka může obsahovat více druhů zboží, je nutné rozložit tento vztah typu M:N a vložit mezi tyto entity další entitu vybrané zboží. Tato entita nám vztah M:N rozdělí na jeden vztah typu 1:M mezi zbožím a vybraným zbožím a druhý vztah typu 1:M mezi objednávkou a vybraným zbožím. Dále zde je entitní množina přepravce, který zákazníkovi dopraví zboží domů. Jeden přepravce může přepravovat zboží k více objednávkám, avšak jedna objednávka může být přepravována jen jedním přepravcem. Poslední entitní množinou je způsob platby, jelikož zákazník si může zvolit, jak chce za objednávku zaplatit. Zde je kardinalita vztahu stejná jako u přepravce.

6.4 Popis entit a jejich atributů

6.4.1 Objednávka

V tabulce objednavka je atribut id_objednavky, který slouží jako jednoznačný identifikátor, a proto je primárním klíčem. Dále zde jsou atributy zakaznik, zamestnanec, zpusob_platby a prepravce, které jsou cizími klíči a odkazují se na tabulky zakaznik, zamestnanec, zpusob_platby a prepravce. Všechny jsou datového typu char. Dalšími atributy jsou datum_prijeti, datum_vyrizeni a datum_dodani, které jsou datem vytvoření, datem vyřízení a datem dodání a jsou datového typu date. A posledními atributy jsou cena_zbozi, cena_platby, cena_dopravy, které jsou datového typu number a znázorňují ceny za danou položku. Atributy datum_vyrizeni a datum_dodani jsou definovány jako null, protože nemusí být objednávka v daný moment vyřízena ani doručena a termíny nejsou známy. Ostatní atributy tabulky objednavka jsou definovány jako not null, jelikož jsou pro vedení informací o objednávce nezbytné a v čase vytvoření jsou známy.

objednavka			
id_objednavky	char	pk	not null
zakaznik	char	fk	not null
zamestnanec	char	fk	not null
prepravce	char	fk	not null
zpusob_platby	char	fk	not null
datum_prijeti	date		not null
datum_vyrizeni	date		null
datum_dodani	date		null
cena_zbozi	number		not null
cena_platby	number		not null
cena_dopravy	number		not null

Tabulka 7 - Objednávka

6.4.2 Zákazník

Tabulka zakaznik obsahuje atribut id_zakaznika, který slouží jako primární klíč. Atributy jmeno, prijmeni, psc, tel, e_mail, login a heslo jsou datového typu char a definovány jsou jako not null. Slouží k evidenci informací o zákazníkovi, jak jejich název napovídá. U atributů ulice a mesto je zvolen datový typ varchar, jelikož některé české ulice a města mají dlouhý název, a proto je varchar vhodnějším řešením a jsou definovány také jako not null. Atributy titul_pred_jmenem a titul_za_jmenem jsou datového typu char a definovány jsou jako null, jelikož nejsou vyžadovány.

zakaznik			
id_zakaznika	char	pk	not null
jmeno	char		not null
prijmeni	char		not null
titul_pred_jmenem	char		null
titul_za_jmenem	char		null
ulice	varchar		not null
město	varchar		not null
psc	char		not null
tel	char		not null
e_mail	char		not null
login	char		not null
heslo	char		not null

Tabulka 8 - Zákazník

6.4.3 Zaměstnanec

Tabulka zamestnanec má jako svůj primární klíč atribut id_zamestnance. Dále jsou zde atributy rodne_cislo a cislo_bu, které mají datový typ char a jsou definovány jako

not null. Upřesňují nám informace o zaměstnanci a to v podobě jeho rodného čísla a čísla jeho bankovního účtu. Všechny ostatní atributy jsou shodné jako u tabulky zakaznik.

zamestnanec			
id_zamestnance	char	pk	not null
jmeno	char		not null
prijmeni	char		not null
titul_pred_jmenem	char		null
titul_za_jmenem	char		null
rodne_cislo	char		not null
ulice	varchar		not null
město	varchar		not null
psc	char		not null
tel	char		not null
cislo_bu	char		not null

Tabulka 9 - Zaměstnanec

6.4.4 Zboží

Tabulka zbozi má jako primarní klíč atribut id_zboží. Dalším atributem je nazev, který je datového typu char a popis, který je datového typu varchar a oba jsou definovány jako not null. Posledním atributem je cena_s_dph, která je datového typu number a je rovněž vyžadována.

zbozi			
id_zbozi	char	pk	not null
nazev	char		not null
popis	varchar		not null
cena_s_dph	number		not null

Tabulka 10 - Zboží

6.4.5 Vybrané zboží

Tabulka vybrane_zbozi je tzv. vazební tabulkou, což znamená, že rozděluje vztah M:N mezi tabulkami zbozi a objednavka. Vazební tabulka je používána v případě, že je možná vícenásobná kombinace řádků z první tabulky s řádky z druhé tabulky. Jelikož jedna objednávka může obsahovat několik různých druhů zboží a zároveň jeden druh zboží může být ve více objednávkách, je nutno využít vazební tabulky. Proto jsou v tabulce vybrane_zbozi cizí klíče id_objednavky a id_zbozi, které tvoří složený primární klíč. Dalšími atributy jsou cena a mnozstvi, které jsou datového typu number. Všechny atributy jsou definovány jako not null.

vybrane_zbozi			
id_objednavky	char	pfk	not null
id_zbozi	char	pfk	not null
cena	number		not null
mnozstvi	number		not null

Tabulka 11 - Vybrané zboží

6.4.6 Způsob platby

Tabulka zpusob_platby má za primarní klíč atribut id_platby. Dalším atributem je nazev_platby, datového typu char a posledním atributem cena_platby, datového typu number. Atribut nazev_platby musí být jedinečný, ostatní jsou vyžadovány.

zpusob_platby			
id_platby	char	pk	not null
nazev_platby	char		unique
cena_platby	number		not null

Tabulka 12 - Způsob platby

6.4.7 Přepravce

Tabulka prepravce má za primární klíč atribut id_prepravce. Druhým atributem je nazev_prepravce, datového typu char. Posledním atributem je cena_prepravy, datového typu number. Všechny atributy jsou vyžadovány.

prepravce			
id_prepravce	char	pk	not null
nazev_prepravce	char		not null
cena_prepravy	number		not null

Tabulka 13 - Přepravce

6.5 Návrh

Jelikož práce je zaměřena na integritu dat, bude zde na zvoleném příkladu demonstrováno správné naprogramování databáze z předešlého příkladu. Je to jen malá část celého programování databázového systému.

6.5.1 Postup návrhu

Vytvořený ER model se převede do množiny relačních tabulek, které byly popsány výše. V nich se definují první integritní omezení, jako například primární a cizí klíče, ale i typy atributů a jejich možný rozsah. V ER modelu a množině relačních tabulek se objevují první doménové, entitní a referenční integrity. Při definici dat se využívá

příkazů DDL, jako například CREATE TABLE nebo ALTER TABLE. Následným postupem je návrh zbývajících omezení integrity, pokud jsou nutné, kterými jsou triggery.

6.5.2 Omezení integrity a jejich využití

Omezení integrity přesně definují data, která lze do databáze vložit, a tím zajišťují správnost dat v ní uložených. Nevhodně napsaná nebo zcela špatná omezení mohou poškodit správnost uložených dat, proto je nutné, aby byly vytvořeny správně, zcela v souladu s požadavky kladenými na databázi.

```
CREATE TABLE objednavka (
    id_objednavky CHAR(10) PRIMARY KEY NOT NULL,
    zakaznik CHAR(10) NOT NULL,
    zamestnanec CHAR(5) NOT NULL,
    prepravce CHAR(3) NOT NULL,
    zpusob_platby CHAR(3) NOT NULL,
    datum_prijeti DATE NOT NULL,
    datum_vyrizeni DATE NULL,
    datum_doruceni DATE NULL,
    cena_zbozi NUMBER(999999) NOT NULL,
    cena_platby NUMBER(999) NOT NULL,
    cena_dopravy NUMBER(999) NOT NULL);
```

Uvedeným kódem příkazu CREATE TABLE byla vytvořena tabulka s objednávkami. Je zde zajištění entitní integrity primárním klíčem id_objednavky. Ten zajišťuje jednoznačnou identifikaci každého řádku v tabulce, a tím znemožňuje vložit další záznam se stejným identifikátorem.

```
INSERT INTO objednavka VALUES ('2015123456', 'zk12345678', 'zm123', 'pr2', 'zp1', '2015-01-20', '2015-01-21', '2015-01-25', 15999, 0, 129);
```

Uvedeným kódem příkazu INSERT INTO byl vložen záznam jedné objednávky do tabulky a to s daty: id_objednavky (primárním klíčem) 2015123456, zakaznik zk12345678, zamestnanec zm123, prepravce pr2, zpusob_platby zp1, datum_prijeti 20.1.2015, datum_vyrizeni 21.1.2015, datum_dodani 25.1.2015, cena_zbozi 15999, cena_platby 0 a cena_dopravy 129.

```
INSERT INTO objednavka VALUES ('2015123456', 'zk87654321', 'zm213', 'pr1', 'zp2', '2015-03-3', '2015-03-5', '2015-03-7', 8999, 30, 149);
```

Pokud se ovšem pokusíme vložit druhý záznam se stejným primárním klíčem, ale rozdílnými daty, databáze tento záznam odmítne, protože dva záznamy nemohou mít stejný primární klíč (identifikátor řádku).

```
CREATE TABLE vybrane_zbozi (
    id_objednavky CHAR(10) NOT NULL,
    id_zbozi CHAR(6) NOT NULL,
    cena NUMBER(999999) NOT NULL,
    mnozstvi NUMBER(999) CHECK (mnozstvi > 0) NOT NULL,
    CONSTRAINT pk_vyb_zb PRIMARY KEY (id_objednavky, id_zbozi),
    FOREIGN KEY (id_objednavky) REFERENCES objednavka(id_objednavky),
    FOREIGN KEY (id_zbozi) REFERENCES zbozi(id_zbozi));
```

Příkaz na vytvoření tabulky vybrane_zbozi, která obsahuje složený primární klíč, který se skládá z atributů id_objednavky a id_zbozi, avšak lze použít i pro jednoduchý primární klíč. Zároveň jsou tyto atributy cizími klíči, které odkazují na tabulky objednavka a zbozi.

```
ALTER TABLE objednavka ADD (
    CONSTRAINT zak_obj FOREIGN KEY (zakaznik) REFERENCES zakaznik(id_zakaznika),
```

CONSTRAINT zam_obj FOREIGN KEY (zamestnanec) REFERENCES zamestnanec(id_zamestnance),

CONSTRAINT prepr_obj FOREIGN KEY (prepravce) REFERENCES prepravce(id_prepravce),

CONSTRAINT zp_pl_obj FOREIGN KEY (zpusob_platby) REFERENCES zpusob_platby(id_platby));

Pomocí ALTER TABLE byly nadefinovány cizí klíče v tabulce objednavka, které tvoří referenční integritu mezi tabulkou objednavka a tabulkami zakaznik, zamestnanec, prepravce a zpusob_platby. Jako příklad zvolím vztah mezi tabulkami prepravce a objednavka. Tabulka prepravce odkazuje do tabulky objednavka, protože jeden přepravce může přepravovat několik objednávek, ale jedna objednávka může mít pouze jednoho přepravce. Vztah mezi nimi je 1:N, tudíž v tabulce objednavka bude definován cizí klíč pojmenovaný prepr_obj.

INSERT INTO prepravce VALUES ('pr1', 'Česká pošta', 149);

INSERT INTO objednavka VALUES ('2015123456', 'zk87654321', 'zm213', 'pr1', 'zp2', '2015-03-3', 8999, 30, 149);

INSERT INTO objednavka VALUES ('2015456789', 'zk12345678', 'zm123', 'pr2', 'zp1', '2015-01-20', 15999, 0, 129);

Jestliže se pokusíme vytvořit objednávku, ke které neexistuje přepravce, nebude vytvořena. V prvním případě s přepravcem pr1, který byl do databáze vložen, nebude při vytváření objednávky problém. Ovšem v druhém případě nebyl přepravce pr2 vložen do databáze, a proto nebude možné objednávku vytvořit, jelikož bylo porušeno integritní omezení pro nenalezení klíče přepravce pr2.

mnozstvi NUMBER(999) CHECK (mnozstvi > 0) NOT NULL;

Přípustné hodnoty pro daný atribut zajišťuje doménová integrita. Ta je určena především datovým typem každého atributu (char, number, date, atd.). To znamená, že pokud máme například datový typ DATE pro datum, tak do něj lze uložit jen platné kalendářní datum a nic jiného. Pokud máme dvě čísla datového typu NUMBER

a ty od sebe odečteme, získáme jako výsledek další číslo. V tabulce vybrane_zbozi je dále doménová integrita zajištěna klauzulí CHECK u atributu mnozstvi, a to tak že omezuje přípustnou množinu čísel, které musí být větší než 0.

mnozstvi NUMBER(999) CHECK (mnozstvi BETWEEN 1 AND 10) NOT NULL;

Takto by vypadal příkaz CHECK, kdybychom měli omezené množství daného zboží na objednávku v rozmezí 1 až 10 kusů.

CREATE TABLE zbozi (

id_zbozi CHAR(6) PRIMARY KEY NOT NULL,

nazev CHAR(30) NOT NULL,

popis VARCHAR NULL,

cena_s_dph NUMBER(999999) NOT NULL);

Dalším možným omezením v tabulce je NOT NULL, které máme například v tabulce zbozi u atributů id_zbozi, nazvu a cena_s_dph a zajišťuje nám vyplnění hodnoty daného atributu při tvorbě záznamu. Opakem je omezení NULL, které máme v tabulce zboží u atributu popis. Toto omezení nám povoluje v atributu hodnotu null a může zůstat při naplňování tabulky prázdné. U tabulky zbozi byly zvoleny hodnoty NOT NULL takto, jelikož atribut id_zbozi (primární klíč) musí být uveden pro přesnou identifikaci záznamu, u atributu nazev z nutnosti rozeznání druhu zboží a u atributu cena_s_dph, protože musíme znát cenu, za kterou zboží prodáváme. Atribut popis má definováno omezení NULL, protože je možné, že se vyskytne zboží, které bude dostatečně charakterizováno svým názvem a nebude třeba již dále specifikovat popis.

CREATE TABLE zpusob_platby (

id_platby CHAR(3) PRIMARY KEY NOT NULL,

nazev_platby CHAR(25) UNIQUE,

cena_platby NUMBER(999) NOT NULL);

Omezení UNIQUE v tabulce zpusob_platby u atributu nazev_platby nám zaručí, že všechny hodnoty jsou navzájem různé. Například nemůžeme mít dvakrát vložený název platby dobírka, i kdyby měl každý jiný primární klíč. V případě vkládání způsobů plateb bychom obdrželi při druhém vkládání chybovou hlášku s tím, že bylo porušeno omezení unikátnosti.

Zhodnocení

Bakalářská práce měla za cíl objasnit problematiku ohledně uplatnění integritních omezení v relačně databázovém zpracování. Mezi cíle v teoretické části patřilo vymezení základních pojmů kolem relačních databází a především datové integrity. Z této části vyplynulo, že je důležité kvalitně analyzovat daný systém, pečlivě rozvrhnou entity a jejich vztahy a nakonec navrhnout správné integritní omezení, které nebudou způsobovat nekorektnosti v databázi.

V následující kapitole byl vybrán centrální registr vozidel, jako databázový systém, na kterém byly demonstrovány nedostatky současných řešení v podobě relačních databází. Z podrobného prozkoumání několika zdrojů objasňujících problematiku kolem registru, vyplynuly nedostatky v navrhovaném řešení realizovaného firmou ATS-Telcom jako jeho dodavatele. Proto byla navržena zjednodušená forma, v níž byly popsány jednotlivé entity a atributy, která byla následně pomocí jazyka SQL vytvořena včetně autorem navržených integritních omezení, jež eliminují nedostatky dodaného řešení, což bylo v zápětí otestováno na několika příkladech. Poslední částí kapitoly bylo zhodnocení problematiky centrálního registru vozidel a argumentů dodavatelské firmy, jež mělo nezávisle posoudit, kde nejspíše vznikly chyby, s nimiž se registr potýká či potýkal.

Poslední kapitolou byl návrh vlastního řešení na fiktivním subjektu pro demonstraci nabytých vědomostí a realizaci korektního návrhu integritních omezení. Pro práci byl zvolen zjednodušený příklad internetového obchodu, který ovšem vycházel z reálné situace. Jako první důležitý krok byl správný návrh ER modelu, aby odpovídal zadání a definovaným potřebám. Tato část je zlomovým okamžikem mezi teoretickým návrhem a praktickou realizací a proto je nezbytné postupovat pečlivě při jeho návrhu. Dalším krokem byl podrobný popis entit a atributů včetně integritních omezení, které se jich týkaly. Zde vyplynulo, že je nezbytné důkladně promyslet situaci kolem každého atributu v dané entitě, aby se předešlo chybám, které by mohly být ve finálním řešení fatální. Posledním krokem bylo pomocí příkazů jazyka SQL vytvořit některé entity včetně jejich integritních omezení a zdůvodnit na základě demonstrace některých příkazů jejich význam v daném řešení. Byly zde využity integritní omezení primary key, foreign key, not null, unique a check.

Závěr

Na základě požadavků kladených na dnešní zpracovávání dat jsou relační databáze vhodným nástrojem pro tuto činnost. Relační databáze obsahuje silné nástroje pro udržení datové integrity, která je nezbytná pro správnou realizaci dané problematiky do databázového zpracování, a jazyk SQL je pro tuto práci spolehlivým a ověřeným nástrojem. Přínos integrity dat spočívá v přesnosti, úplnosti a konzistentnosti databáze. Avšak v současné době jsou integritní omezení často nedostatečně využívány, nejspíše z důvodu opomíjení fáze plánování a nevhodně či nedostatečně stanoveným cílům a dalším detailům, které blíže specifikují daný databázový systém. Doménová integrita je tím, co zajišťuje důkladnější omezení jednotlivých atributů, ale bez dobrého základu ve formě referenční a entitní integrity není výsledný celek omezení integrity kvalitní, a databáze tak nemusí odrážet skutečnost, kterou by měla. Práce demonstrovala možnosti využití těchto tří typů omezení, které jsou stěžejní pro každou databázi. Další možností by byla ještě omezení pokročilejším způsobem za pomoci triggerů, které jsou nejnáročnější na realizaci a tato práce je ve své praktické části neobsahuje. Závěrem práce tedy je, že čím kvalitněji jsou navrženy integritní omezení, tím kvalitnější je i výsledný databázový systém a ten odráží realizovanou problematiku o to lépe, což by měl být záměr každého databázového systému.

Seznam literatury

[1] HERNANDEZ, Michael J. *Návrh databází*. 1. vyd. Praha: Grada, 2006, 408 s. ISBN 80-247-0900-7.

[2] OPPEL, Andrew J. *SQL bez předchozích znalostí: [průvodce pro samouky]*. Vyd. 1. Brno: Computer Press, 2008, 240 s. ISBN 978-80-251-1707-1.

[3] SKŘIVAN, Jaromír. Databáze a jazyk SQL. [online]. [cit. 2014-10-21]. Dostupné z: <http://interval.cz/clanky/databaze-a-jazyk-sql/>

[4] Triggery. [online]. [cit. 2014-11-10]. Dostupné z: <http://www.602.cz/datainc/produkty/winbase/napoveda/html/sql237jt.htm>

[5] Nový registr vozidel obsahuje skoro půl milionu chyb v údaji o majiteli vozidla. [online]. [cit. 2015-01-03]. Dostupné z: <http://www.dnoviny.cz/silnicni-doprava/novy-registr-vozidel-obsahuje-skoro-pul-milionu-chyb-v-udaji-o-majiteli-vozidla>

[6] VOSTROVSKÝ, Václav. *Vytváření databází v ORACLE*. Vyd. 1. V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta, 2004, 132 s. ISBN 978-80-213-1191-6.

[7] KYTE, Thomas. *Oracle: návrh a tvorba aplikací*. Vyd. 1. Brno: CP Books, 2005, 694 s. Programování. ISBN 80-251-0569-5.

[8] CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Vyd. 1. Brno: Computer Press, 2009, 584 s. ISBN 978-80-251-2328-7.

[9] JAROSLAV, Pokorný. *Databázové systémy*. Vyd. 2. Praha: ČVUT, 2004, 148 s. ISBN 80-010-2789-9.

Seznam tabulek

Seznam obrázků