Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Datová integrita v relačně databázovém zpracování

Roman Kapek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Roman Kapek

Informatika

Název práce

Datová integrita v relačně databázovém zpracování

Název anglicky

Data integrity in a relational database processing

Cíle práce

Bakalářská práce je zaměřena na problematiku využití datové integrity v databázově koncipovaných informačních zabezpečení podnikatelských subjektů. Hlavním cílem této práce je:
a) objasnit teoretické principy relačně databázové technologie a to především datové integrity v kontextu s informačním zabezpečení podnikatelských subjektů,
b) zmapovat momentální stav této problematiky a vymezit její relevantnost včetně požadavků na ni kladených,
c) navrhnout potencionální možnosti řešení této problematiky v souladu s identifikovanými požadavky,
d) ověřit funkčnost navržených záležitostí,
e) ověřené záležitosti zobecnit pro další možná uplatnění.

Metodika

Použitá metodika zadané bakalářské práce bude založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Stěžejními metodami této práce budou metody a techniky relačně databázové technologie. 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.

Závazný harmonogram:

Vymezení teoretických principů řešené problematiky, literární rešerše – do 4.9.2016: předmět 1. zápočtu z BP,

Zmapování současné situace řešené problematiky – do 15. 11. 2016

Navržení konkrétního řešení – do 25.1.2017: předmět 2. zápočtu z BP

Ověření a zobecnění navrhovaných záležitostí – do 14.3.2017: předmět 3. zápočtu z BP.

Doporučený rozsah práce

45-55stran

Klíčová slova

Relačně databázová technologie, datová integrita, informační zabezpečení, SQL

Doporučené zdroje informací

Begg,C.,Conolly,T.,Holowczak,R.: Mistrovství databáze, profesionální průvodce tvorbou efekvních databází. Computer press. Brno 2009. ISSBN 978-80-251-2328-7.

Bryla,B.,Loney,K.: Mistrovství v Oracle Database 11g. Computer press. Brno 2009. ISSBN 978-80-251-2189-4.

Molinaro,A.: SQL, kuchařka programátora. Computer press. Brno 2009. ISSBN 978-80-251-2617-2.

Valenta, M., Pokorný, J.: Databázové systémy. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

Předběžný termín obhajoby

2016/17LS – PEF

Vedoucí práce

doc. Dr. Ing. Václav Vostrovský

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2016

Ing.Marn Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2016

Ing.Marn Pelikán, Ph.D.

Děkan

V Praze dne 19. 02. 2017

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 použitých zdrojů 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 13.3.2017 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Václavu Vostrovskému, Ph.D. za vedení této práce a za jeho cenné rady a připomínky ke zlepšení úrovně této bakalářské práce.

Datová integrita v relačně databázovém zpracování

Souhrn

Bakalářská práce je zaměřena na problematiku využití datové integrity v relačních databázích. Teoretická část obsahuje vymezení důležitých pojmů ohledně databází, relačních databází, jazyka SQL a především datové integrity v rámci tohoto jazyka.

Druhá část práce obsahuje výběr současného databázového řešení z běžného života, dále obsahuje analýzu jeho nedostatků a návrh správného řešení. Na závěr této kapitoly je zhodnoceno dané databázové řešení.

V poslední části je realizováno vlastní řešení, které je zaměřeno primárně na demonstraci správného návrhu integritních omezení. Tato část obsahuje správný návrh ER diagramu, který musí vyhovovat zadaným cílům a potřebám. Dále tato část obsahuje popis tabulek a jejich atributů. A na závěr je za pomoci jazyka SQL demonstrováno ověření integritních omezení.

Klíčová slova: relační databázová technologie, datová integrita, informační zabezpečení, SQL

Data integrity in a relational database processing

Summary

Bachelor thesis is focused on the use of data integrity in a relational database. The theoretical part contains definitions of important concepts regarding databases, relational databases, SQL and especially data integrity in the context of the SQL.

The second part contains a selection of the current database solution from everyday life, also includes an analysis of its shortcomings and suggest the correct solution. At the conclusion of this chapter there is evaluation of the database solution.

The last part contains an implementation of custom solution that is focused primarily on the demonstration of the proper draft of integrity constraints. This part contains the correct draft of ER diagram, which must meet specified objectives and needs. Additionally, this part includes a description of tables and their attributes. Finally, it is demonstrated verification of integrity constraints with using precepts of SQL language.

Keywords: relational database technology, data integrity, information security, SQL

Obsah

1 Úvod

Zpracování dat se dnes prolíná do všech oborů v rámci lidské činnosti. Dnešní moderní společnost je postavena na databázových systémech (evidence občanů, zdravotnictví, hospodářství atd..). Za pomoci výpočetní techniky jsou sbírána data a z nich člověk získává informace, které pomáhají k jeho rozhodování. Oblast nasazování databázových systému je rok od roku širší a na ukládání, vyhledávání a prezentaci dat za pomoci počítače a jeho programového vybavení je kladen stále větší důraz.

Databáze vznikají kvůli potřebě změnit data na informace. Data jsou surová (nezpracovaná) fakta, která mají určitou důležitost pro jednotlivce nebo organizaci. Informace jsou data, která prošla zpracováním nebo dostala strukturu, která jim dává pro jednotlivce nebo organizaci význam. Např. jména a čísla v telefonním seznamu jsou pouze data, ale číslo konkrétní osoby je už informace.

Datová integrita je téma, které je aktuální, potřebné a velice využívané již mnoho let v aplikacích a databázích. Vývoj databázových technologií během posledních několika let vedl ke vzniku výkonnějších databázových systémů, které je možné ovládat intuitivněji. Ve skutečnosti jsou v současnosti databázové systémy tak rozšířené a tvoří tak nedílnou součást našeho každodenního života, že si často ani nejsme vědomi, že nějaký databázový systém používáme. V současnosti většina organizací používá nějaký databázový systém ke správě a zabezpečení jejich dat. Také na internetu přibývá velké množství dat, která je potřeba zabezpečit proti zneužití. Za pomoci správné integrity těchto dat a dalších prvků zabezpečení lze zajistit potřebné zabezpečení pro tyto data. Tato práce s daty bývá často přehlížená a nedoceněná. V praxi je význam ochrany/zabezpečení dat podceňován a lidé věří ve schopnosti jim dostupných technologií. Reálná hodnota dat bývá jejich vlastníkem doceněna až při jejich neautorizovaném využití, zničení nebo momentální, či trvalé nedostupnosti. Proto je důležité se touto problematikovou zabývat. Při návrhu databáze je integrita dat jednou z hlavních složek, aby byl daný návrh databáze kvalitní. Bez datové integrity nelze spolehlivě zpracovávat a uchovávat data.

V České republice toto téma souvisí s problémy v registru vozidel, který je znám přetrvávajícími problémy a špatným zpracováním. Například problémům s nemožností

identifikace majitele vozu nebo problémům se špatně vloženou identifikací majitele vozidla by se dalo předejít za pomoci správných a s rozvahou použitých integritních omezení.

Téma datové integrity je aktuální v databázích používaných ve všech oborech lidské činnosti (např. nákup v obchodním domě, návštěva knihovny, půjčovna DVD atd..). Správně zpracovaná a zabezpečená data velice ulehčují vyhledávání a to hlavně při vyhledávání ve velkém množství dat. Správně navržená integritní omezení usnadňují práci všem lidem, kteří budou s danými daty pracovat a zamezí poškození nebo vkládání nevhodných dat.

Teoretická část této bakalářské práce popisuje pojmy a principy relační databáze a to především datové integrity a jejího využití. Rozebírá integritní omezení entitní, doménové, referenční, dále triggery a popisuje jejich využití. Zabývá se pojmy databáze, relační databáze, dotazovací jazyk SQL, tabulka, databázové klíče a vazby mezi tabulkami. Dále popisuje momentální stav dané problematiky a obsahuje využití a řešení integritních omezení pomocí dotazovacího jazyka SQL.

2 Cíl práce a metodika

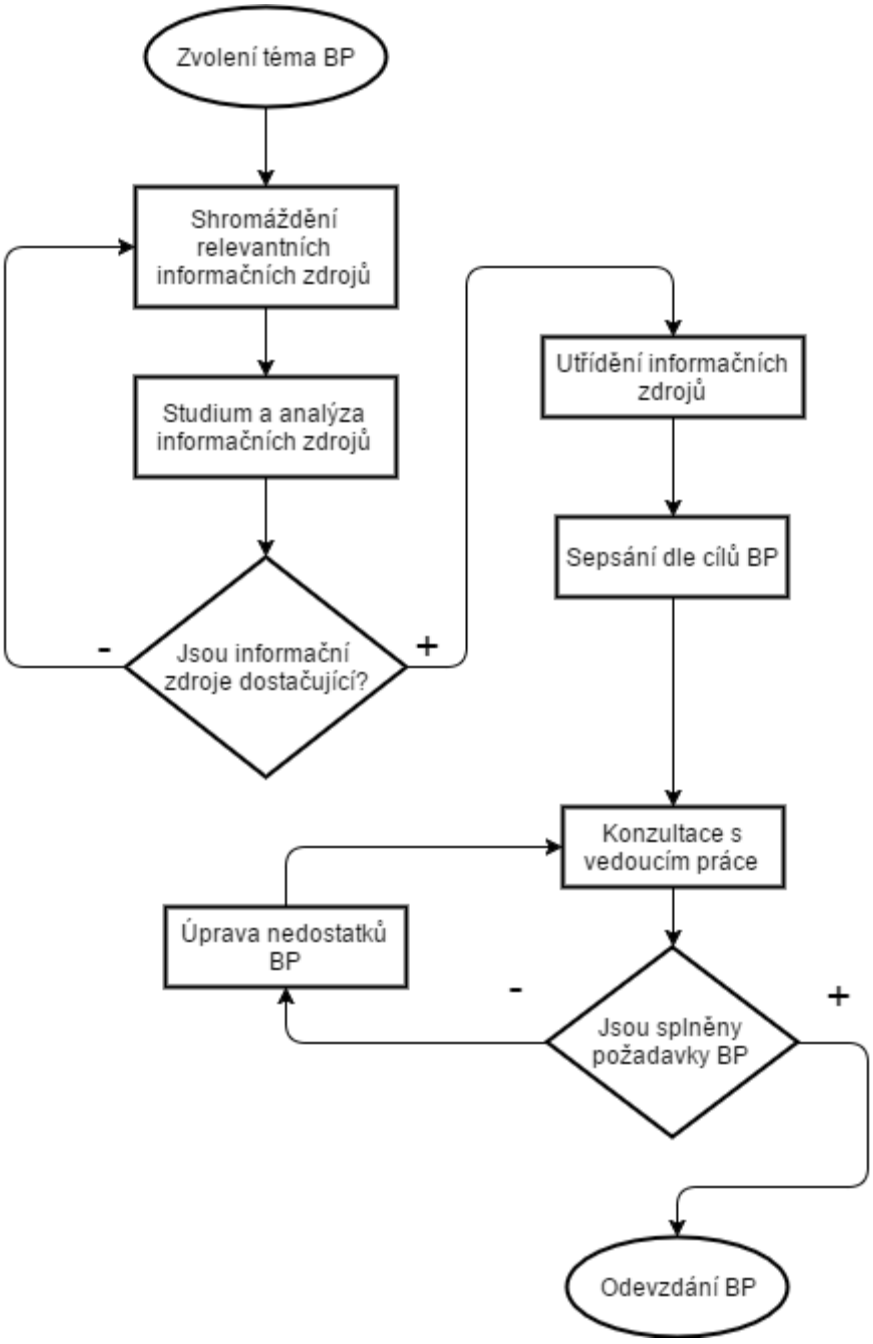
2.1 Cíl práce

Tato bakalářská práce je zaměřena na problematiku využití datové integrity v rámci relačních databází. Hlavním cílem této práce je:
a) Analýza současného databázového řešení z běžného života a správný návrh, ve kterém jsou odstraněny jeho chyby a nedostatky.
b) Návrh vlastního řešení zaměřeného především na datovou integritu, v tomto řešení bude demonstrován správný návrh databáze, veškeré kroky budou objasněny a budou zde uvedeny příkazy, které jsou nutné pro vytvoření integritních omezení, které tento návrh obsahuje.

2.2 Metodika

Tato bakalářská práce je vytvořena na základě studia vhodné odborné literatury a vhodných online dostupných zdrojů, které se zabývají relačně databázovou technologií, informačním zabezpečením, datovou integritou a jazykem SQL. Získané informace z těchto zdrojů jsou následně analyzovány, utříděny a postupně sepsány podle cílů této práce. Nejdříve je sepsána teoretická část, následně je sepsána část, která osahuje výběr aktuálního databázového řešení, jeho analýzu, zhodnocení a navrhnutí funkčního řešení. A na konec je navrženo vlastní řešení, které je vytvořeno primárně z důvodů demonstrace příkazů pracujících s integritními omezeními. Příkazy SQL, které jsou obsaženy v této práci jsou vytvořeny na bázi databázového systému Oracle.

Vývojový diagram:



Obrázek 1: Vývojový diagram metodiky Zdroj: Vlastní tvorba

3 Teoretické principy řešené problematiky

3.1 Databáze

Databáze tak, jak je známe dnes, začali vznikat v 50. letech dvacátého století. Avšak uchovávání a třídění dat bylo potřeba mnohem dříve. Za první doloženou zmínku o ucelené snaze uchovávat data lze považovat knihovnu v Ugaritu (město na území dnešní Sýrie). V Ugaritu se nalezlo pohromadě větší množství hliněných tabulek (diplomatický text a literární tvorba) pocházející již z 12. století před našem letopočtem. Data se zde začala sbírat, nikoliv třídit. Snaha o třídění dat se objevila až u knihovny v římském Foru Romanu. [9]

Až kartotéka byla předchůdcem počítačových databází. Kartotéka umožňovala uspořádávat data podle různých kritérií a zařazovat nové položky. Veškeré operace s ní prováděl přímo člověk a její správa se velice podobala správě dnešních databází. V dnešní době je většina kartoték nahrazena relačními databázemi. Ke komunikaci s relačními databázemi slouží jazyk SQL, který je standardizovaný institutem ANSI. Prakticky každá relační databáze podporuje tyto standardy. Proto je vysoce přenosný mezi různými typy relačních databází. [9]

Pojem databáze je v dnešní době, v oblasti počítačů hojně používaný. Běžně se označením databáze myslí jak uložená data, tak i software. Důležité je, že databázi obvykle nevlastní žádné oddělení nebo uživatel, ale je sdíleným zdrojem společnosti. Je to místo, kam se určitým způsobem ukládají organizované a strukturované údaje. Všechny tyto údaje jsou integrovány s minimálním množstvím duplikací. Přístup k těmto údajům obstarává databázový systém, kterému se říká SŘBD - Systém Řízení Báze Dat (anglicky - DBMS - DataBase Managment System). Za databázi lze považovat telefonní seznamy, jízdní řády, seznam knih v knihovně, seznam prodávaného zboží v obchodě nebo seznam pacientů u doktora. Tyto informace jsou shromažďovány a počítačově evidovány za účelem jejich praktického využití. Hlavním důvodem pro zakládání databází je jejich rychlé a spolehlivé vyhledávání požadovaných informací. Databáze nahradili své předchůdce kartotékové systémy, se kterými jsme se mohli setkat v knihovnách (karty knih), skladech (skladové karty) nebo u lékaře (karty pacientů). V případě velkého množství údajů bylo u těchto kartotékových systémů složité aktualizovat jednotlivé údaje nebo vyhledávat a uspořádávat karty podle různých požadavků. [3,10]

Databázi si lze představit jako soubor dat, který slouží pro popis reálného světa (např. evidence školní knihovny, sklad chemikálií, evidence studentů). Entita (tabulka) je prvek reálného světa (např. člověk, město, kniha), který je popsán svými vlastnostmi. Tyto vlastnosti se většinou považují za atribut (např. jméno,plat, název, věk). [4]

3.2 **Relační databáze**

Relační databáze se poprvé objevila v roce 1969 jako dílo Dr. Edgara F. Codda, ten navrhl možnost, jak použít relační kalkul a algebru i pro netechnické uživatele při ukládání a manipulaci s daty. Podle Codda k tomu mělo být používáno srozumitelných příkazů, které vycházely z angličtiny. Tento návrh už předpokládal ukládání dat do tabulek.
Tabulky je možné mezi sebou kombinovat jako běžné množiny za pomoci základních operací (sjednocení, rozdíl, spojení, průnik a kartézský součin). [6]

Z důvodů své dobré pochopitelnosti a jednoduchosti se relační databáze hojně využívají. Nejznámějšími zástupci relačních databázových systému jsou Oracle, MySQL, MS SQL, a další. [5]

Základem relačních databází jsou databázové tabulky. Tyto tabulky jsou mezi sebou určitým způsobem závislé - existuje mezi nimi jistá logická vazba (relace). Tabulky se nazývají také entity a jsou chápány jako prvek reálného světa (např. žák, učitel, kniha, automobil). [5]

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

Tabulka má jméno, které ji odlišuje od všech ostatních tabulek v příslušné databázi.
Každá buňka tabulky obsahuje přesně jednu hodnotu. Například by byla chyba ukládat několik telefonních čísel pro distribuční centrum do jedné buňky. Jinými slovy tabulky neobsahují opakují se skupiny dat. O relační tabulce, která splňuje tuto vlastnost, říkáme, že byla normalizována do první normální formy.
Každý sloupec má jedinečné jméno.
Všechny hodnoty v jednom sloupci jsou stejné domény.

Pořadí sloupců nemá význam. Jinak řečeno, pokud se sloupec přesune spolu se svými hodnotami, můžeme sloupec vzájemně zaměňovat.
Každý záznam je jedinečný; neexistují duplicitní záznamy.
Pořadí záznamů nemá význam, teoreticky. Ale v praxi jejich pořadí může ovlivnit efektivitu přístupu k záznamům. [10]

3.3 **Jazyk SQL**

SQL je nejrozšířenějším komerčním databázovým jazykem, navržen byl pro odborníky i neodborníky. Historie jazyka SQL spadá do 70. a 80. let. První standard, který byl přijat v roce 1986 se označoval SQL86. Časem se však projevili některé nedostatky a v roce 1992 vyšla opravená verze, která se označovala SQL92. Ta je v oblasti relačních databází standardem dodnes. Zkratka SQL označuje Structured Query language. Jazyk v sobě zahrnuje jak nástroje pro tvorbu databází (tabulek), tak nástroje pro manipulaci s daty (vkládání dat, aktualizaci, mazání a vyhledávání informací). [4,10]

SQL se skládá z několika částí, některé jsou určeny pro administrátory a návrháře databázových systémů, jiné pak pro koncové uživatele a programátory. První část jazyka SQL je jazyk DDL - Data Definition Language. DDL je jazyk pro vytváření databázových schémat a katalogů. Další část je jazyk SDL - Storage Definition Language. SDL definuje způsob ukládání tabulek. Třetí částí pro návrháře a správce je jazyk VDL - View Definition Language, ten určuje vytváření pohledů (pohled si lze představit jako virtuální tabulku složenou z různých jiných tabulek.) Poslední částí SQL je jazyk DML - Data Manipulation Language, který obsahuje základní příkazy (INSERT, UPDATE, DELETE a SELECT). S jazykem DML nejvíce pracují koncoví uživatelé a programátoři. [4]

3.3.1 **Využitelnost SQL**

V ideálním případě by měl uživateli umožnit:
Vytvářet struktury databází a tabulek.
Provádět údržbu dat, například vkládání, modifikaci a vymazání dat z tabulek.
Provádět jednoduché i složité dotazy. [10]

3.3.2 **Výhody SQL**

Lze se ho relativně snadno naučit.
Jedná se neprocedurální jazyk: je třeba zadat, jaké informace požadujete, nikoli jak je získat
Má volný formát, což znamená, že části příkazů není nutné psát na přesně stanovené místo na obrazovce.
Struktura dotazu je založena na standardních anglických slovech jako SELECT (vyber), INSERT (vlož), UPDATE (aktualizuj) a DELETE (vymaž).
SQL může být implementován mnoha uživateli včetně administrátora databáze, řídících pracovníků, aplikačních programátorů a koncových uživatelů. [10]

3.4 **Základní pojmy**

3.4.1 **Tabulka**

Každá tabulka je tvořena sloupci a řádky, sloupce tabulky představují vlastnosti této entity (např. pro knihu - název,cena, nakladatelství, ISBN, žánr atd.). Tyto vlastnosti se nazývají atributy. Každý sloupec musí mít jedinečný název a datový typ podle dat, které jsou v něm uložena (řetězec znaků, čísla, datum a čas, atd.).
Řádky tabulky představují samotné záznamy v databázové tabulce. Jeden řádek tabulky představuje např. jednu knihu s hodnotami daných vlastností - atributů (autor, žánr,..) a každý řádek by také měl obsahovat svůj jedinečný identifikátor, podle něj bude možné vybrat příslušný záznam. [5]

ID_knihy	nazev_knihy	cena	zanr
1	Krteček a myška	120,00 Kč	kreslený
2	Krteček ve vesmíru	200,00 Kč	věda a technika

Tabulka 1: Tabulková struktura Zdroj: Vlastní tvorba

3.4.2 **Klíče relace**

Každý záznam v tabulce musí být jedinečný. To znamená, že musíme být schopni určit sloupce nebo kombinaci sloupců (klíče relace), které tuto jedinečnost zajišťují. [10]

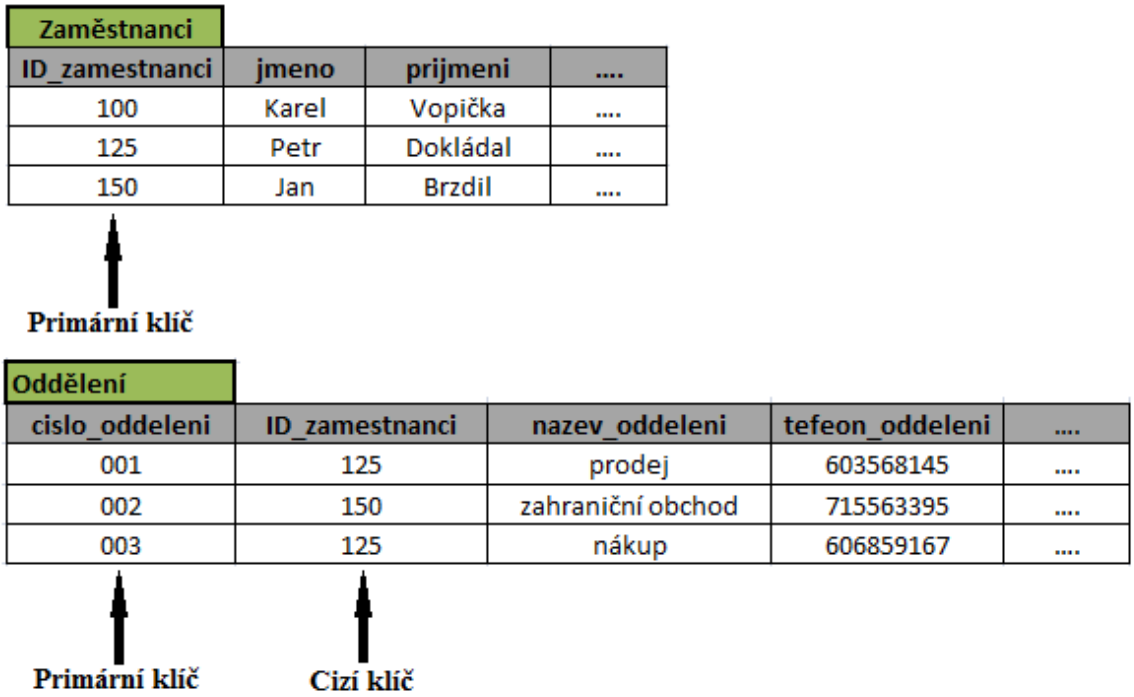
Super klíč - je to sloupec nebo množina sloupců, které jedinečně identifikují záznam v relaci, může obsahovat nadbytečné sloupce

Kandidátní klíč - je super klíč, který obsahuje jen minimální počet sloupců nutných k jedinečné identifikaci záznamů a pro tabulku má dvě vlastnosti

- Jedinečnost - v každém záznamu jeho hodnota určuje výlučně daný záznam.
- Neredukovatelnost - žádná jeho vlastní podmnožina nezajišťuje jedinečné určení záznamů.

Primární klíč - je kandidátní klíč, který je vybrán, aby jedinečně určoval záznamy v tabulce

Cizí klíč - jedná se o sloupce nebo skupinu sloupců v jedné tabulce, která odpovídá kandidátnímu klíči některé tabulky [10]



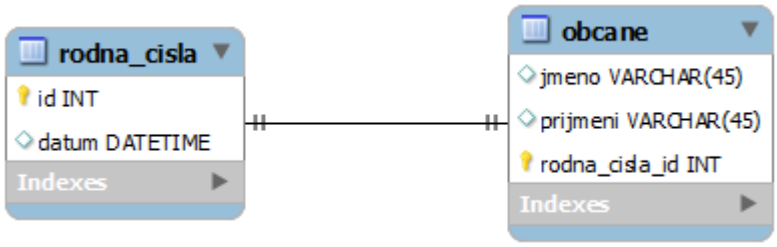
Obrázek 2: Primární a cizí klíč Zdroj: Vlastní tvorba

3.4.3 **Vazby mezi tabulkami**

Vazby mezi tabulkami slouží ke spojení dat, která spolu souvisejí a jsou umístěny v různých databázových tabulkách. Vazba mezi tabulkami může být realizována několika způsoby. Záleží na vztahu informací v jedné tabulce, k informacím v jiné tabulce. [3,5]

3.4.3.1 Vazba 1:1

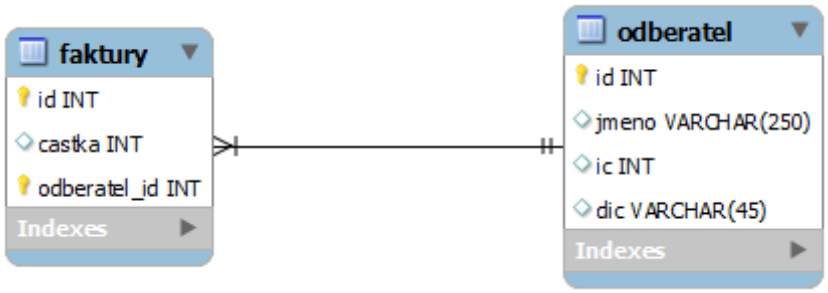
Jedná se o spojení, kdy jedné položce z první tabulky odpovídá právě jedna položka z druhé tabulky. Na obrázku č. 3 jednomu rodnému číslu odpovídá právě jeden občan. [5]



Obrázek 3: Vazba 1:1 Zdroj: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

3.4.3.2 Vazba 1:N

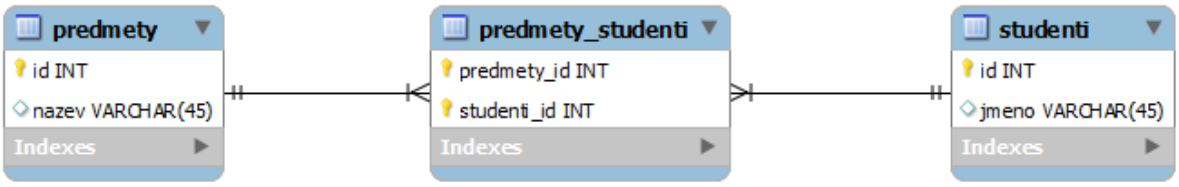
U vazby 1:N se jedná o spojení, kde každé položce z první tabulky odpovídá N-položek z tabulky druhé. Na obrázku č. 4 každý odběratel může mít více faktur, ale naopak každá faktura má právě jednoho odběratele. [5]



Obrázek 4: Vazba 1:N Zdroj: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

3.4.3.3 Vazba M:N

U této vazby M-položkám z první tabulky odpovídá N-položek z druhé tabulky. Na obrázku č. 5 každý předmět může mít více studentů a zároveň každý student může studovat více předmětů. Tento vztah bývá nejčastěji realizován kombinací dvou vztahů 1:N a 1:M, které ukazují do vazební (pomocné) tabulky. [5]



Obrázek 5: Vazba M:N Zdroj: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

3.5 Integrita dat

Relační databáze musí obsahovat prostředky pro zajištění ochrany dat, které obsahuje před možným úmyslným, ale i neúmyslným poškozením dat a také před jejich zneužitím. Už při návrhu a vytváření databáze je zapotřebí zachytit všechna pravidla, která umožní v databázi zajistit správnost (integritu) uložených dat. Integrita dat je zajištění kompletnosti (úplnosti) a správnosti uložených dat v databázi. To znamená, že data uložená v databázi jsou konzistentní vůči definovaným pravidlům. Je možné zadávat pouze data, která vyhovují těmto předem definovaným pravidlům. Tato pravidla se označují pojmem integritní omezení (integrity constraints). Integritní omezení jsou nástroje, které zabraňují vkládání nesprávných dat či ztrátě nebo poškození stávajících záznamů v průběhu práce s databází. Jsou to omezení, které musíme zavést, proto aby se databáze nestala neúplnou, nepřesnou nebo nekonzistentní. Například je možné zajistit mazání dat, která již ztratila svůj význam - např. pokud smažeme uživatele, tak se odstraní i zbytek jeho záznamů v ostatních databázových tabulkách. [3,10]

Integrita dat se dá považovat za další formu ochrany databáze. Týká se zabezpečení, ale má širší důsledky. Integrita se týká hlavně kvality dat samotných. Obvykle se vyjadřuje v již zmíněných termínech omezení, což jsou pravidla konzistence, která není dovoleno porušit. Omezení se mohou vztahovat na data v jednom záznamu nebo se mohou týkat vztahů mezi záznamy. [10]

Datová integrita se nesmí podceňovat nebo přehlížet, to by vedlo ke vzniku chyb, které se dají pouze velice těžko najít a odstranit. Pokud by došlo ke vzniku takovýchto chyb, tak se stane to, že dojde k rozhodování na základě špatných informací. [10]

3.5.1 **Druhy integritních omezení**

3.5.1.1 Entitní integrita

Entitní integrita se vztahuje na primární klíče tabulek. Primární klíč je minimální identifikátor, který se užívá k jedinečnému určení záznamů. To znamená, že žádná podmnožina primárního klíče nestačí k jedinečnému určení záznamů. Pokud bychom povolili prázdnou hodnotu pro jakoukoli část primárního klíče, znamenalo by to, že ne všechny sloupce jsou potřebné k rozlišení mezi záznamy, což odporuje definici primárního klíče. Například protože nějaký sloupec je primárním klíčem tabulky, nemůžeme do tabulky vložit záznam s prázdnou hodnotou v tomto sloupci. [10]

3.5.1.2 Referenční integrita

Referenční integrita se vztahuje na cizí klíče. Je to nástroj, který pomáhá udržovat vztahy mezi záznamy v relačně propojených tabulkách. Definuje se jako vztah mezi dvojicí tabulek. Tabulka ve které je pravidlo referenční integrity uvedeno, se nazývá podřízená tabulka a tabulka, jejíž jméno je v integritním omezení uvedeno, se nazývá tabulka nadřízená. V obou těchto tabulkách je určen sloupec nebo skupina sloupců, které tvoří klíč. Pravidlo referenční integrity požaduje, aby pro každý záznam v tabulce podřízené, jehož klíč nemá hodnotu NULL, existoval v nadřízené tabulce záznam se stejným klíčem. [2,10]

3.5.1.3 Doménová integrita

Doménová integrita se vztahuje na přípustné hodnoty daného atributu (sloupce) relační tabulky. Toto omezení definuje přípustné hodnoty pro tyto atributy. Pro zachování doménové integrity je důležité zajistit, aby daný sloupec obsahoval pouze data pro tento sloupec přípustná. Například může být požadováno, aby plat byl v rozmezí 8000 - 50000 Kč. [3,10]

3.5.2 **Dodržování integritních omezení**

3.5.2.1 Mechanismy na straně databázového serveru

Nejlepší způsob z hlediska ochrany dat.
Uživateli obvykle přináší delší odezvu systému a nelze pokaždé zajistit jejich přenositelnost na jiný databázový systém.

3.5.2.2 Mechanismy na straně klienta

Nejlepší volba pro komfort a nezávislost na databázovém systému.
Nutnost kontrolních mechanismů pro každou operaci, kvůli možnosti chyby v aplikaci.

3.5.2.3 Samostatné programové moduly na straně serveru

V moderních databázových systémech jsou pro tento účel implementovány triggery, které lze spouštět automaticky před a po operacích manipulujících s daty
Umožňuje implementaci i složitých integritních omezení.
Malá možnost přenesení na jiný databázový systém. [3,10]

3.5.3 **Triggery**

V řadě informačních systémů, které běží nad nějakou databází, je potřeba v případě vzniku nějaké události (např. modifikace řádku v tabulce), automaticky spustit příkaz, který provede nějaké operace. Pro tento účel slouží triggery (z angl. trigger = "spoušť"). Triggery jsou mocný nástroj, který usnadňuje práci s daty. [7]

Triggery zjednodušují tvorbu aplikaci, protože přenášejí část práce databázové aplikace na server. Např. pokud v podnikovém informačním systému existuje tabulka zaměstnanců, lze za pomocí triggerů popsat všechny akce, které mají být provedeny při přijetí, propuštění zaměstnance, nebo změně platu atd. [8]

Vlastnosti triggerů:
Je svázán s určitou tabulkou a reaguje na změny v této tabulce.
Reaguje na právě jednu z SQL akcí INSERT, DELETE nebo UPDATE.
Provádí se před (BEFORE), nebo po (AFTER) provedení výše specifikované akce.

Nelze vyvolat editací multiatributu.
Pokud akce ovlivňuje více než jeden záznam, tak se může spouštět pro každý ovlivněný záznam zvlášť nebo pro všechny záznamy najednou.
Může obsahovat podmínku, která se vyhodnotí před provedením triggeru a pokud nebude splněna, trigger se nespustí.
Specifikuje akci, která bude automaticky provedena, pokud jsou splněny podmínku pro spuštění triggeru. [8]

4 Současná situace řešené problematiky

4.1 Centrální registr vozidel

K demonstraci řešené problematiky byl jako příklad z praxe vybrán centrální registr vozidel. Ten je znám, po přechodu na nový systém, který zprostředkovávala firma ATS-Telcom svými přetrvávajícími problémy. Chyby, které vznikly jsou způsobené jednak amatérským přístupem, a dále tím, že firma ATS-Telcom neměla dostatek zkušeností.

4.2 Chyby v centrálním registru vozidel

Dopravní noviny 27. července 2012

„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. " [11]

4.3 Příčiny chyb v centrálním registru vozidel

V případě 497 640 záznamů s chybně vloženou identifikací majitele vozidla nastala chyba ve špatně nastavené referenční integritě. To má za následek, že se v systému nezobrazuje aktuální majitel vozidla, ale předchozí majitel. A nastává problém například, když Česká kancelář pojistitelů (ČKP) rozesílá výzvy k uhrazení povinného ručení majitelům motorových vozidel, kteří tuto zákonem danou povinnost ignorují. Ale daná osoba již konkrétní motorové vozidlo nevlastní. Dále v případě stíhání majitele vozidla, z důvodu přestupku se může bývalý vlastník dostat do problémů, místo pravého majitele i přesto, že konkrétní vozidlo již nevlastní.

U 414 888 záznamů, kde není možné majitele identifikovat při srovnávání s dalšími databázemi nastala chyba opět ve špatně nastavené referenční integritě, ale i v doménové integritě, to znamená, že se do databáze majitelů dostala nesprávná data. A znovu vniká problém při posílání výzvy k uhrazení povinného ručení, nebo při stíhání majitele vozidla, z důvodu přestupku.

V téměř 21 000 případech, kde nesouhlasí registrační značka došlo k chybě ve špatně nastavené referenční integritě, což má za následek, že se u vozidel v systému nezobrazuje správná registrační značka.

V 7000 případech, kde se v databázi objevily duplicitní VIN kódy došlo jednoznačně k porušení entitní integrity. VIN kód je unikátní číslo pro každé vozidlo, to znamená, že duplicita VIN kódů v databázi je závažná chyba. Například mohlo být vydáno na jeden VIN kód několik technických průkazů na více různých jmen.

4.4 Návrh zjednodušeného řešení centrálního registru vozidel

4.4.1 Tabulky

Tabulka registrační značka

registracni_znacka			
reg_znac	char	primary key	not null
datum_vyd	date		null
kraj	char		not null

Tabulka 2: Registrační značka Zdroj: Vlastní tvorba

Tabulka majitel

majitel			
rodne_cislo	char	primary key	not null
jmeno	char		not null
prijmeni	char		not null
psc	char		not null
mesto	char		not null
ulice	char		not null

Tabulka 3: Majitel Zdroj: Vlastní tvorba

Tabulka vozidlo

vozidlo			
vin	char	primary key	not null
reg_znac	char	foreign key	unique not null
rodne_cislo	char	foreign key	not null
rok_vyroby	number		not null
vyrobce	char		not null
model	char		not null
barva	char		not null

Tabulka 4: Vozidlo Zdroj: Vlastní tvorba

4.4.2 **Vytvoření tabulek**

Registrační značka

```
SQL> CREATE TABLE registracni_znacka
  2  (reg_znac CHAR(7) PRIMARY KEY NOT NULL,
  3  datum_vyd DATE NULL,
  4  kraj CHAR(20) NOT NULL);

Table created.
```

Majitel

```
SQL> CREATE TABLE majitel
  2  (rodne_cislo CHAR(11) PRIMARY KEY NOT NULL,
  3  jmeno CHAR(20) NOT NULL,
  4  prijmeni CHAR(20) NOT NULL,
  5  psc char(6) NOT NULL,
  6  mesto CHAR(35) NOT NULL,
  7  ulice CHAR(35) NOT NULL);

Table created.
```

Vozidlo

```
SQL> CREATE TABLE vozidlo
  2  (vin CHAR(17) PRIMARY KEY NOT NULL,
  3  reg_znac CHAR(7) UNIQUE NOT NULL,
  4  rodne_cislo CHAR(11) NOT NULL,
  5  rok_vyroby NUMBER(4) NOT NULL,
  6  vyrobce CHAR(15) NOT NULL,
  7  model CHAR(15) NOT NULL,
  8  barva CHAR(10) NOT NULL,
  9  FOREIGN KEY (reg_znac) REFERENCES registracni_znacka(reg_znac),
 10  FOREIGN KEY (rodne_cislo) REFERENCES majitel(rodne_cislo)
 11  );

Table created.
```

4.4.3 **Naplnění tabulek hodnotami**

Naplnění tabulky registrační značka

```
SQL> INSERT INTO registracni_znacka
  2  VALUES ('2L02404','22.4.2001','Liberecky');

1 row created.

SQL> INSERT INTO registracni_znacka
  2  VALUES ('7S59790','18.8.2005','Stredocesky')

1 row created.

SQL> INSERT INTO registracni_znacka
  2  VALUES ('5S53223','3.11.2007','Stredocesky');

1 row created.

SQL> INSERT INTO registracni_znacka
  2  VALUES ('2P18899','5.4.2010','Plzensky');

1 row created.
```

Naplnění tabulky majitel

```
SQL> INSERT INTO majitel
  2  VALUES ('800313/4502','Marek','Vesely','350 42','Tuchlovice','Zamkova');

1 row created.

SQL> INSERT INTO majitel
  2  VALUES ('901119/8878','Karel','Machacek','251 34','Lhota','Prazska');

1 row created.

SQL> INSERT INTO majitel
  2  VALUES ('845829/8871','Katerina','Mala','314 27','Plzen','Nerudova');

1 row created.
```

Naplnění tabulky vozidlo

```
SQL> INSERT INTO vozidlo
  2  VALUES ('3HSCDAXN63N020328','2L02404','845829/8871',1998,
  3  'Skoda','Fabia','Stribrna');

1 row created.

SQL> INSERT INTO vozidlo
  2  VALUES ('5TFLV52149X093689','7S59790','800313/4502',2004,
  3  'Skoda','Octavia','Zelena');

1 row created.

SQL> INSERT INTO vozidlo
  2  VALUES ('2GCFC29K6R1220603','5S53223','901119/8878',2005,
  3  'Chevrolet','Cruze','Cerna');

1 row created.
```

4.4.4 Testování vytvořených tabulek

Duplicita VIN kódu

Pokud je vkládán duplicitní vin kód, který je v tabulce **vozidlo** nastaven jako primární klíč, nastane chyba. A tak nemůže dojít k tomu, že by se v databázi objevilo více shodných vin kódů.

```
SQL> INSERT INTO vozidlo
  2  VALUES ('2GCFC29K6R1220603','6S31254','921115/5239',2004,
  3  'Chevrolet','Captiva','Bila');
INSERT INTO vozidlo
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C008911) violated
```

Registrační značka u více vozidel

Při vkládání již použité registrační značky k dalšímu vozidlu dojde ke stejné chybě jako v předchozím případě. Protože v tabulce **vozidlo** je u atributu **reg_znac** nastaveno omezení unique. A to je proto, že reálně není možné, aby více aut mělo shodnou registrační značku.

Jeden majitel u více aut

Pokud je vytvořeno vozidlo s majitelem, který již má přidělené jiné vozidlo, tak nedojde k žádnému problému. Toto musí být povoleno, protože reálně je možné,aby jeden člověk byl majitelem více vozidel.

```
SQL> INSERT INTO vozidlo
  2  VALUES ('1MEBP81F9BZ677938','2P18899','845829/8871','2000',
  3  'Audi','A8','Modra');

1 row created.
```

4.4.5 **Závěrečné zhodnocení současné situace**

Na zjednodušeném řešení je ukázáno, jak lze pomocí správně navržených integritních omezení předejít chybám, které nastaly v databázi registru vozidel.

Obchodní ředitel firmy ATS-Telcom, který je dodavatelem nové aplikace pro registr vozidel, Martin Tůma, se v médiích opakovaně vyjádřil, že společnost neudělala žádnou chybu. Ale chyby, které v databázi registru vozidel vznikly jsou jednoznačně chybou této firmy. Pravděpodobně se tak stalo, protože firma v té době neměla dostatek zkušeností a integritní omezení podcenila a špatně navrhnula. [12]

5 Praktická část

5.1 Charakteristika projektu

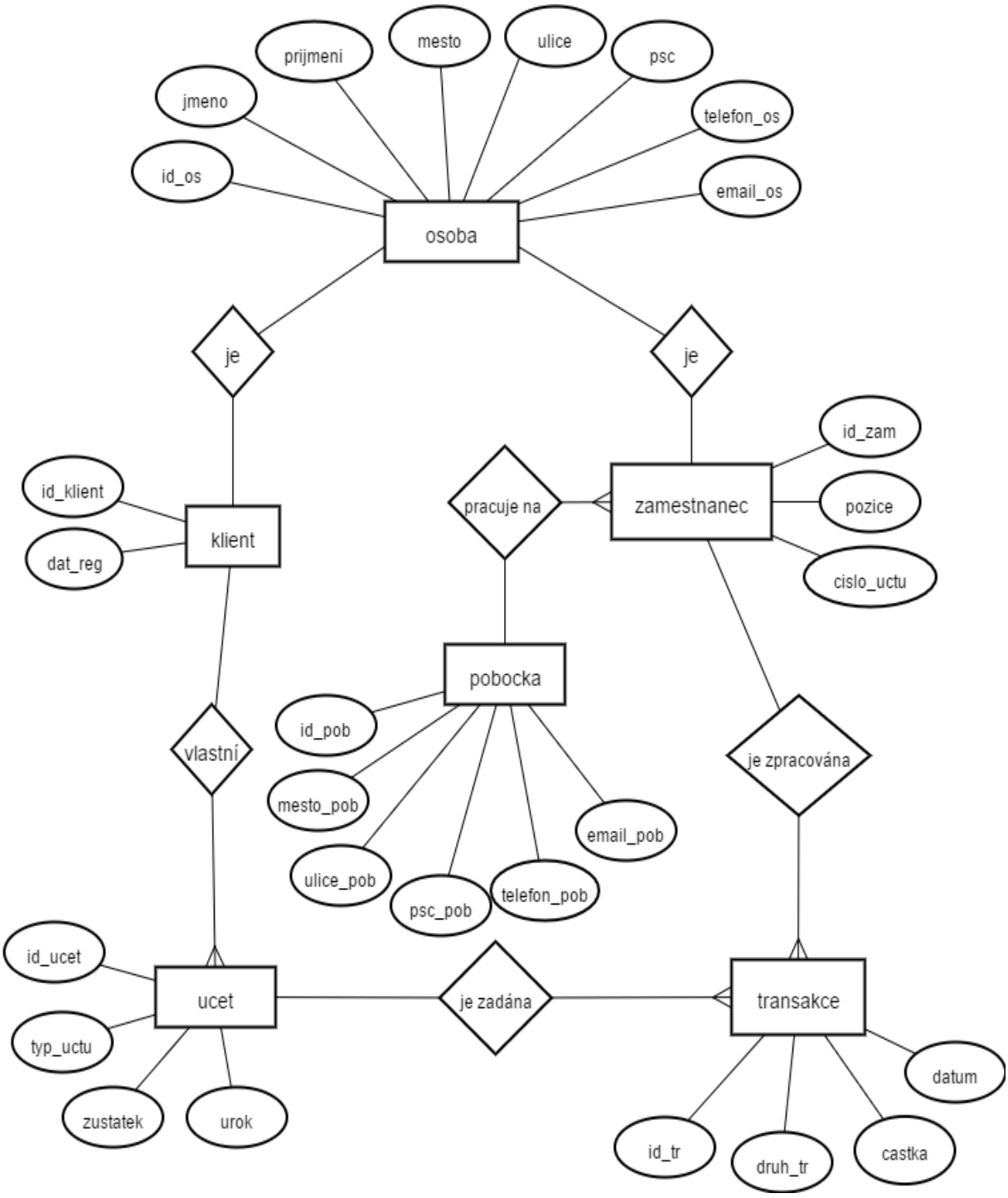
K demonstraci správného návrhu a využití integritních omezení v relačních databázích byl zvolen jednoduchý model databáze banky. Většina lidí vlastní u banky nějaký účet a umí s ním manipulovat, tak by tento projekt neměl být náročný pro pochopení.

Tento model databáze banky slouží pro evidování jednotlivých klientů banky, zaměstnanců banky, účtů, pro evidování transakcí, které v bance proběhli, a také pro evidenci poboček banky.

Nejdůležitější jsou informace o proběhnutých transakcích. Pro banku je důležité mít možnost zpětně dohledat informace o tom, který klient, kdy a jakou transakci provedl, a také který zaměstnanec danou transakci provedl v případě, kdy nastane jakýkoliv problém.

Tento projekt je zjednodušen pro demonstraci správného návrhu a využití integritních omezení. Ve skutečnosti by databáze byla mnohem složitější.

5.2 **E-R diagram**



Obrázek 6: ER diagram Zdroj: Vlastní tvorba

5.3 Vztahy v E-R diagramu

E-R diagram je tvořen entitami osoba,klient, zaměstnanec, pobočka, účet, transakce a dále jejich atributy a vztahy mezi nimi. Jeden klient je právě jednou osobou a zároveň osoba je jedním klientem. To samé platí pro vztah mezi zaměstnancem a osobou. Zaměstnanec pracuje na jedné pobočce, ale každá pobočka může mít více zaměstnanců. Účet vlastní jeden klient, ale každý klient může být vlastníkem více účtů. Transakce je zadána z jednoho účtu, ale z každého účtu může být zadáno více transakcí. Zaměstnanec může zpracovávat více transakcí, ale každá z transakcí je zpracována pouze jedním zaměstnancem.

5.4 Popis tabulek a jejich atributů

5.4.1 Osoba

osoba			
id_os	char	primary key	not null
jmeno	char		not null
prijmeni	char		not null
mesto	char		not null
ulice	char		not null
psc	char		not null
telefon_os	char		not null
email_os	char		not null

Tabulka 5: Osoba Zdroj: Vlastní tvorba

V tabulce **osoba** jako primární klíč slouží atribut id_os, který má datový typ char. Další atributy jméno, příjmení, město, ulice, psč, telefon_os a email_os jsou také datového typu char. Všechny atributy v této tabulce jsou definovány jako not null, protože je potřeba všechny znát. Jméno, příjmení, město, ulici, psč, aby bylo jasné o koho se jedná a kde bydlí. A telefon s emailem ke komunikaci.

5.4.2 **Klient**

klient			
id_klient	char	primary key	not null
id_os	char	foreign key	not null
dat_reg	date		not null

Tabulka 6: Klient Zdroj: Vlastní tvorba

Jako primární klíč v tabulce **klient** slouží atribut id_klient. Dále tabulka obsahuje atribut id_os, který je cizím klíčem ukazující do tabulky osoba. Atribut dat_reg je datum registrace klienta u banky a má datový typ date. Všechny atributy v této tabulce jsou definovány jako not null.

5.4.3 **Pobočka**

pobocka			
id_pob	char	primary key	not null
mesto_pob	char		not null
ulice_pob	char		not null
psc_pob	char		not null
telefon_pob	char		not null
email_pob	char		not null

Tabulka 7: Pobočka Zdroj: Vlastní tvorba

V této tabulce **pobočka** je primárním klíčem atribut id_pob, má datový typ char. Ostatní atributy mají také datový typ char. Všechny tyto atributy jsou definovány jako not null, protože je potřeba všechny znát. Město, ulici, psč, aby bylo jasné, kde pobočka sídlí. A tel s emailem ke komunikaci.

5.4.4 **Zaměstnanec**

zamestnanec			
id_zam	char	primary key	not null
id_os	char	foreign key	not null
id_pob	char	foreign key	not null
pozice	char		not null
cislo_uctu	char		not null

Tabulka 8: Zaměstnanec Zdroj: Vlastní tvorba

V tabulce **zaměstnanec** slouží jako primární klíč atribut id_zam, který má datový typ char. Další atributy id_os a id_pob mají datový typ char a jsou cizími klíči, které ukazují do tabulek osoba a pobočka. Ostatní atributy jsou také datového typu char. Všechny atributy z této tabulky jsou definovány jako not null.

5.4.5 **Účet**

ucet			
id_ucet	char	primary key	not null
id_klient	char	foreign key	not null
typ_uctu	char		not null
zustatek	number		not null
urok	char		not null

Tabulka 9: Účet Zdroj: Vlastní tvorba

Primárním klíčem v tabulce **účet** je atribut id_účet a má datový typ char. Dále je v tabulce atribut id_klient datového typu char, který je cizím klíčem a ukazuje do tabulky klient. Atributy typ_účtu a úrok jsou také datového typu char a atribut zůstatek je datový typ number. Všechny tyto atributy jsou definovány jako not null.

5.4.6 **Transakce**

transakce			
id_tr	char	primary key	not null
id_ucet	char	foreign key	not null
id_zam	char	foreign key	not null
druh_tr	char		not null
castka	number		not null
datum	date		not null

Tabulka 10: Transakce Zdroj: Vlastní tvorba

V tabulce **transakce** je primárním klíčem atribut id_tr s datovým typem char. Atributy id_účet a id_zam jsou datového typu char a jsou cizími klíči ukazujícími do tabulek účet a zaměstnanec. Atribut druh_tr je datového typu char, částka datového typu number a datum datového typu date. Všechny atributy jsou definovány jako not null.

5.5 **Vytvoření tabulek a integritních omezení**

Vytvořením integritních omezení se přesně definuje, která data do databáze patří a zajistí se, aby nedošlo ke ztrátě dat, která se ztratit nemají. Data jsou konzistentní pokud jsou ve vztahu vyhovující integritním omezením. To znamená, že se žádnou úpravou dat (úpravou, smazáním) neztratila nebo nepoškodila data, nebo že v DB nejsou data, která tam nemají co dělat, například kontakty na smazaného klienta atp. Proto existují integritní omezení, která mají za úkol podobným nehodám zabránit.

5.5.1 **Osoba**

```
SQL> CREATE TABLE osoba
  2  (id_os CHAR(12) PRIMARY KEY NOT NULL,
  3  jmeno CHAR(10) NOT NULL,
  4  prijmeni CHAR(15) NOT NULL,
  5  mesto CHAR(15) NOT NULL,
  6  ulice CHAR(15) NOT NULL,
  7  psc CHAR(6) NOT NULL,
  8  telefon_os CHAR(15) NOT NULL,
  9  email_os CHAR(25) NOT NULL);

Table created.
```

Vytvoření tabulky **osoba** s primárním klíčem id_os.

5.5.2 **Klient**

```
SQL> CREATE TABLE klient
  2  (id_klient CHAR(12) PRIMARY KEY NOT NULL,
  3  id_os CHAR(12) NOT NULL,
  4  dat_reg DATE NOT NULL);

Table created.

SQL> ALTER TABLE klient ADD
  2  (CONSTRAINT fk_osoba FOREIGN KEY (id_os)
  3  REFERENCES osoba(id_os));

Table altered.
```

Vytvoření tabulky **klient** s primárním klíčem id_klient a cizím klíčem id_os ukazujícím do tabulky **osoba**.

5.5.3 **Pobočka**

```
SQL> CREATE TABLE pobocka
  2  (id_pob CHAR(12) PRIMARY KEY NOT NULL,
  3  mesto_pob CHAR(15) NOT NULL,
  4  ulice_pob CHAR(15) NOT NULL,
  5  psc_pob CHAR(6) NOT NULL,
  6  telefon_pob CHAR(15) NOT NULL,
  7  email_pob CHAR(25) NOT NULL);

Table created.
```

Vytvoření tabulky **pobočka**, která má primární klíč id_pob.

5.5.4 **Zaměstnanec**

```
SQL> CREATE TABLE zamestnanec
  2  (id_zam CHAR(12) PRIMARY KEY NOT NULL,
  3  id_os CHAR(12) NOT NULL,
  4  id_pob CHAR(12) NOT NULL,
  5  pozice CHAR(15) NOT NULL,
  6  cislo_uctu CHAR(20) NOT NULL);

Table created.

SQL> ALTER TABLE zamestnanec ADD
  2  (CONSTRAINT fk_osoba_zam FOREIGN KEY (id_os)
  3  REFERENCES osoba(id_os),
  4  CONSTRAINT fk_pobocka FOREIGN KEY (id_pob)
  5  REFERENCES pobocka(id_pob));

Table altered.
```

Vytvoření tabulky **zaměstnanec** s primárním klíčem id_zam a cizími klíči id_os a id_pob ukazujícími do tabulek **osoba** a **pobočka**.

5.5.5 **Účet**

```
SQL> CREATE TABLE ucet
  2  (id_ucet CHAR(20) PRIMARY KEY NOT NULL,
  3  id_klient CHAR(12) NOT NULL,
  4  typ_uctu CHAR(10) NOT NULL,
  5  zustatek NUMBER(10) NOT NULL,
  6  urok CHAR(5) NOT NULL);

Table created.

SQL> ALTER TABLE ucet ADD
  2  (CONSTRAINT fk_klient FOREIGN KEY (id_klient)
  3  REFERENCES klient(id_klient));

Table altered.
```

Vytvoření tabulky **účet**, která má primární klíč atribut id_účet a cizí klíč id_klient, který ukazuje do tabulky **klient**.

5.5.6 **Transakce**

```
SQL> CREATE TABLE transakce
  2  (id_tr CHAR(12) PRIMARY KEY NOT NULL,
  3  id_ucet CHAR(20) NOT NULL,
  4  id_zam CHAR(12) NOT NULL,
  5  druh_tr CHAR(5) NOT NULL,
  6  castka NUMBER(10) NOT NULL,
  7  datum DATE NOT NULL);

Table created.

SQL> ALTER TABLE transakce ADD
  2  (CONSTRAINT fk_ucet FOREIGN KEY (id_ucet)
  3  REFERENCES ucet(id_ucet),
  4  CONSTRAINT fk_zam FOREIGN KEY (id_zam)
  5  REFERENCES zamestnanec(id_zam));

Table altered.
```

Vytvoření tabulky **transakce** s primárním klíčem id_tr a cizími klíči id_účet a id_zam, který ukazují do tabulek **účet** a **zaměstnanec**.

5.6 **Ověření integritních omezení**

5.6.1 **Entitní integrita**

K zajištění entitní integrity slouží primární klíč, ten zaručuje jednoznačné identifikování každého řádku v tabulce. Nelze vložit dva záznamy se stejným primárním klíčem.

Vložení prvního záznamu do tabulky **osoba** s hodnotou 875512/5159 u atributu id_os, který je primárním klíčem proběhne bez problému.

```
SQL> INSERT INTO osoba
  2  VALUES ('875512/5159','Karel','Prachar','Rakovnik','Dolni',
  3  '270 85','702-252-713','karelp1@seznam.cz');

1 row created.
```

Pokud ale dojde k vkládání záznamu do tabulky, který má stejný primární klíč jako již existující záznam, tak databáze zajistí, že dojde k chybě a daný záznam se nevytvoří.

```
SQL> INSERT INTO osoba
  2  VALUES ('875512/5159','Martin','Kolomazna','Praha','Horni',
  3  '280 11','605-805-602','martinkolo@seznam.cz');
INSERT INTO osoba
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C009371) violated
```

5.6.2 **Doménová integrita**

Doménová integrita určuje hodnoty přípustné pro daný atribut. Je zajištěna pomocí datového typu atributu (CHAR, VARCHAR, DATE , NUMBER,..). Například do number lze vkládat pouze čísla, do date pouze platné datum atd. Dále lze doménová integrita zajistit například za pomoci konstrukce CHECK, ta definuje podmínku, kterou musí splňovat každý řádek.

Příklad využití konstrukce CHECK pokud by částka, která lze na účtech převádět omezena do 100000.

```
castka NUMBER(10) CHECK(castka BETWEEN 1 AND 100000) NOT NULL,
```

5.6.3 **Referenční integrita**

K udržení referenční integrity databáze slouží mechanismus omezení cizích klíčů. Jedná se o spojení jednoho nebo více sloupců tabulky se sloupcem nebo více sloupci cizí tabulky.

Pokud bude zadán tento příkaz, bez předchozího vytvoření záznamu osoby s id_os 720423/6897 nebo pobočky s id_pob pob03, tak se tento zaměstnanec nemůže vytvořit. Protože dojde k porušení integritního omezení pro nalezení klíče v tabulce osoba nebo pobočka.

```
SQL> INSERT INTO zamestnanec
  2  VALUES('zam0125','720423/6897','pob03','Vedouci','670105-5626/123');
```

5.6.4 **NULL a NOT NULL**

Omezení NULL a NOT NULL. Klíčové slovo NULL dovoluje, aby sloupec obsahoval hodnoty NULL, to znamená, že může zůstat při vyplňování hodnoty prázdný, zatímco NOT NULL tyto hodnoty zakazuje, zajišťuje vyplnění hodnoty při vytváření sloupce. V SŘBD Oracle platí, že pokud klauzule není uvedena, databáze bude předpokládat hodnotu NULL.

U většiny tabulek jsou všechny atributy definovány jako NOT NULL, protože se jedná o informace, které musí být bezpodmínečně známy.

Například tabulka **osoba**.

```
SQL> CREATE TABLE osoba
  2  (id_os CHAR(12) PRIMARY KEY NOT NULL,
  3  jmeno CHAR(10) NOT NULL,
  4  prijmeni CHAR(15) NOT NULL,
  5  mesto CHAR(15) NOT NULL,
  6  ulice CHAR(15) NOT NULL,
  7  psc CHAR(6) NOT NULL,
  8  telefon_os CHAR(15) NOT NULL,
  9  email_os CHAR(25) NOT NULL);

Table created.
```

Pokud zadáme takovýto příkaz, tak se záznam nevytvoří, protože datum registrace je definován jako NOT NULL a musí být zadán.

```
SQL> INSERT INTO klient
  2  VALUES('kl00589','720423/6897','');
```

5.6.5 **UNIQUE**

Omezení UNIQUE zajišťuje, že všechny hodnoty ve sloupci či množině sloupců jsou navzájem různé.

Takto lze v tabulce **osoba** vytvořit sloupec telefon_os, pokud by bylo požadováno, aby každá osoba vlastnila různé telefonní číslo.

```
telefon_os CHAR(15) UNIQUE NOT NULL,
```

Závěr

Systém s kvalitní databází je stabilní a snadněji udržovatelný. Přínosem integritních omezení je jejich vynucení, už když se databáze vytváří. To znamená, že je zaručena správnost dat už když se databáze vyplňuje poprvé. Pokud je to potřeba, tak je možné vytvořit nová integritní omezení, které se aplikují na již existující data, to přidává možnost dodatečného doplnění integritních omezení nebo úpravy již existujících omezení.

Tato práce demonstrovala aktuální využití doménové, entitní a referenční integrity dat, které jsou důležité při návrhu každé databáze. Jedná se o ukázku využití integrity a zabezpečení přesných a správných dat. Tyto zásady jsou obecně platné a po úpravě pro daný problém jsou použitelné v širším měřítku. Toto řešení tedy lze využít jako návod pro návrh integritních omezení. Pokročilejší způsob omezení lze zajistit za pomoci triggerů, které jsou na realizaci nejnáročnější, ty ale tato práce ve své praktické části neobsahuje.

Korektní a validní data v databázi neulehčují práci pouze programátorům, ale hlavně uživatelům, kteří s danými daty dále pracují. Validní a správná data jsou hlavním důvodem správného fungování celého systému a pokud by data nebyla správná, byla by bezcenná a takovéto data je zbytečné ukládat. Vytváření a používání integritních omezení není jednoduchá záležitost a jsou zapotřebí vysoké kvalifikační požadavky, proto je důležité k jejich tvorbě přistupovat zodpovědně a pozorně. Nejdůležitější jsou znalosti programování, jazyka SQL a databází. Využívání integrity dat přináší své výsledky a tak by měla být součástí každé databáze, protože správná data jsou základem úspěchu.

Seznam použitých zdrojů

[1] Státnice: referenční integrita [online]. [cit. 2016-08-15]. Dostupné z: http://statnice.matfyz.info/generated/ISPS-html/node78.html

[2] 602SQL server: Referenční integrita dvojice tabulek [online]. [cit. 2016-08-15]. Dostupné z: http://sql602.sourceforge.net/helpdir-cs/xml/html/sql_ref_integrity.html

[3] Chytrak: Databáze [online]. [cit. 2016-08-12]. Dostupné z: http://www.databaze.chytrak.cz/

[4] Interval: Databáze a jazyk SQL: Vývoj databází [online]. [cit. 2016-08-12]. Dostupné z: https://www.interval.cz/clanky/databaze-a-jazyk-sql/

[5] Grafická a multimediální laboratoř: Základy relačních databází [online]. [cit. 2016-08-12]. Dostupné z: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

[6] Root: Historie relačních databází [online]. [cit. 2016-08-13]. Dostupné z: http://www.root.cz/clanky/historie-relacnich-databazi/

[7] Interval: SQL - jak na triggery [online]. [cit. 2016-08-16]. Dostupné z: https://www.interval.cz/clanky/sql-jak-na-triggery/

[8] Software602: Triggery [online]. [cit. 2016-08-16]. Dostupné z: http://www.602.cz/datainc/produkty/winbase/napoveda/html/sql237jt.htm

[9] Webová integrace: Historie a trendy ve vývoji databází [online]. [cit. 2016-08-12]. Dostupné z: http://www.web-integration.info/cs/blog/historie-a-trendy-ve-vyvoji-databazi/

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

[11] Dopravní noviny: Chyby v novém registru vozidel [online]. [cit. 2016-10-22]. Dostupné z: http://www.dnoviny.cz/silnicni-doprava/novy-registr-vozidel-obsahuje-skoro-pul-milionu-chyb-v-udaji-o-majiteli-vozidla

[12] Idnes: Problémy jsou u jiných [online]. [cit. 2016-10-29]. Dostupné z: http://zpravy.idnes.cz/martin-tuma-z-ats-telcom-k-registru-vozidel-feo-/domaci.aspx?c=A120726_121729_domaci_abr

[13] BRYLA, Bob a Kevin LONEY. Mistrovství v Oracle Database 11g. Brno: Computer Press, 2009. ISBN 978-80-251-2189-4.

[14] MOLINARO, Anthony. SQL: kuchařka programátora. Brno: Computer Press, 2009. ISBN 978-80-251-2617-2.

[15] POKORNÝ, Jaroslav a Michal VALENTA. Databázové systémy. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

Seznam obrázků

Seznam tabulek