**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**


**Bakalářské práce**

# Integrita dat v relačně databázovém zpracování

**Vladimír Pacejka**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Vladimír Pacejka

Informatika

Název práce

**Integrita dat v relačně databázovém zpracování**

Název anglicky

**Data integrity in 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 této problematiky a identifikovat požadavky s ní spojené,
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 08. 03. 2015

**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci Integrita dat 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 13. 3. 2015 .....

**Poděkování**

Tímto bych rád poděkoval doc. Ing. Václavu Vostrovskému, Ph.D. za cenné rady a připomínky vedoucí ke zlepšení úrovně práce a jejímu zdárnému dokončení. V poslední řadě bych chtěl poděkovat i svým přátelům a rodině.

# Integrita dat v relačně databázovém zpracování

**Souhrn**

Bakalářská práce je tematicky zaměřená na problematiku datové integrity v relačně databázovém zpracování dat. Cílem je vymezit teoretické principy problematiky relačních databází a to především v záležitosti datové integrity a zmapovat současnou situaci a úroveň problematiky. Práce za použití jazyka SQL, databázového nástroje SQL server 2008 a SQL server management studia demonstruje možné použití integritních omezení a business pravidel na příkladu výrobního podniku a využívá SQL dotazy, funkce a triggery. Navržené řešení uspokojuje požadavky pro evidenci výrobků, zaměstnanců, odběratelů, oddělení a objednávek. Řešení prokazuje vhodnost a nutnost použití integritních omezení a lze ho použít jako návod, jak k problematice integrity dat přistupovat.

**Klíčová slova**

Datová základna, datová integrita, relačně databázová technologie, SQL

# Data integrity in relational database processing

**Summary**

This bachelor thesis is thematically focused on problems with data integrity in relational database data processing. The main goal is to define the principles of relational databases and especially the data integrity issue. Also it is mapping the current situation and the level of this issues. The work uses SQL language, dabase tool SQL server 2008 and its SQL server management studio to demonstrate possible usage of the integrity constraints and business rules on the example of a manufacture company. The solution is using SQL queries, functions and triggers and it satisfies the requirements for the evidence of products, employees, departments, customers and orders. Solution demonstrates appropriateness and necessity of using integrity constraints and it could be used as a manual on how to aprroach the data integrity issue.

**Key words**

Data base, data integrity, relational database technology, SQL

# Obsah

# 1 Úvod

Integrita dat je problematikou aktuální, potřebnou a obsáhlou a je využívána již řadu let v aplikacích, databázích a při projektech. V dnešní době je to opět jedno z probíraných témat, zejména v dobách rostoucího množství dat na internetu. Tyto data je třeba chránit proti zneužití a správná integrita těchto dat, napomůže v kombinaci s dalšími prvky ochrany k potřebnému zabezpečení dat. Proto je třeba se touto problematikou systematicky zabývat. Stále se však jedná o často přehlíženou součást práce s daty a bývá zcela nedoceněná. Z pohledu databází je integrita klíčovou složkou návrhu kvalitní databáze a není bez ní možné spolehlivě zpracovávat a uchovávat data.

Předkládaná bakalářská práce se zaměřuje na problematiku integritních omezení v relačně databázovém zpracování. Rozebírá integritu entitní, referenční i doménovou, zmiňuje využití triggerů a business pravidel. Dále pak popisuje užití jednotlivých omezení a jejich výhody. K realizaci integritních omezení využívá jazyka SQL a k uchování databáze byl použit SQL server 2008. Práce je rozdělena na dvě hlavní sekce, jimiž jsou teoretická část a praktická část.

Teoretická část práce vymezuje pojmy a teoretické principy relační databáze, zejména pak problémy a principy spojené s integritou dat a uvádí jejich využití. Zaobírá se transakčním zpracováním a pojmem databáze, jejími principy a technologiemi s ní spojené. Popisuje jednotlivé vztahy v databázi, základní principy tvorby a návrhu databáze. Rozebírá dotazovací jazyk SQL, jednotlivé druhy integrit a jejich užití právě v SQL společně s použitím triggerů a business pravidel. Popisuje též momentální stav dané problematiky. Praktická část názorně ukazuje využití integritních omezení za použití dotazovacího jazyka SQL, kterým jsou realizovány jednotlivé create skripty tabulek a triggery.

Problematika integritních omezení je aktuální v databázích všeho druhu. Kvalitní datová základna je dnes potřebná napříč všemi projekty, které pracují s nějakými daty a napříč všemi podniky, které používají software pro zpracování dat. Relevantní a korektní data usnadňují vyhledávání, zejména při hledání ve větším množství dat a mají kvalitní vypovídací hodnotu. I administrátor může do systému vložit špatná data a tak správné navržení integritních omezení ulehčí práci všem, kteří budou s daty dále pracovat a zabraňuje tak poškození či vložení nevhodných dat. Integrita dat funguje jak v databázi, tak i v aplikaci, která je nástavbou nad danou databází.

V České republice je toto téma aktuální například v souvislosti s projektem centrálního registru vozidel a projektem sociálních dávek, který ukázal názorně mnoho nedostatků. Od

nekonzistentních dat, přes data zcela chybná se tyto projekty nevyhnuli ani naprostým selháním systému. Využitím integritních omezení korektně a s rozvahou by se předešlo mnoha problémům, způsobeným zejména nepřesnými daty a zatížením systému, který byl původně centralizovaný poté převeden na decentralizovaný a dnes je opět převáděn zpět na centralizovaný.

# 2 Cíl práce a metodika

### Cíl práce

Bakalářská práce je tematicky zaměřená na problematiku datové integrity v relačně databázovém zpracování dat. Jejím cílem je vymezení teoretických principů relačních databází, zejména v záležitosti datové integrity, zmapovat současnou situaci této problematiky, navrhnout a ověřit možnosti řešení požadavků na datovou integritu na příkladu z praxe.

### Metodika

Použitá metodika této bakalářské práce byla založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Stěžejní byly metody a techniky relačně databázové technologie ve spojení s problematikou datové integrity. Navrhované řešení zohledňuje požadavky klienta a je spojené s řešenou záležitostí. Na základě syntézy teoretických poznatků a dosažených výsledků jsou formulovány závěry této bakalářské práce.

# 3 Databáze

Již od svého příchodu patřily databáze mezi zkoumané oblasti počítačového odvětví a spolu s databázovými systémy jsou esenciálními komponenty moderního života. Cílem databáze, jakožto úložiště či souboru dat, je podpořit efektivní ukládání, vyhledávání a údržbu dat. V informatice se tedy jedná o data, fyzicky uložená v jednom, nebo ve více souborech na disku, která jsou složena z různých fyzických či logických struktur. Existuje mnoho typů databází, které vyhovují jednotlivým průmyslovým odvětvím. Databáze může být specializována pro ukládání binárních souborů, dokumentů, obrázků či videí a jiných dat, ať již jde o data analytická, transakční nebo geografická a je navrhována, tvořena a plněna daty pro svůj specifický účel. [1]

Databázový přístup ke zpracování dat umožňuje jejich oddělenou definici od jejich údržby uživatelskými programy. Původní souborový přístup ke zpracování dat měl mnoho problémů. Od nekonzistentních izolovaných dat, ke kterým bylo obtížné přistupovat až po problémy s jejich ochranou a integritou a také problém přístupu více uživatelů. Konzistence znamená jednotlivou návaznost efektů transakcí, nenarušenost integritní omezení a provedení validních, korektních a přípustných operací s ohledem na databázovou sémantiku. Databáze zahrnuje datové prvky spolu s vazbami, neboli také vztahy mezi nimi, dále také integritní omezení a schéma a spolu se systémem řízení báze dat se pak nazývá jako celek databázový systém. [2]

Systém řízení báze dat, nebo také anglicky database management system, je softwarové rozhraní mezi aplikačními programy a uloženými daty, které tak zajišťuje jak tvorbu databáze, tak i práci s ní a také její údržbu. Obsahuje dva typy jazyků. Jsou jimi jazyk pro definici dat, který slouží k tvorbě definic dat v databázi, čili jakýsi logický popis dat a jazyk pro manipulaci dat, jenž používáme k aktualizaci a k výběru dat dle požadavků, jakési dotazování. Spolu s pojmem databáze souvisí pojmy data a informace. [2, 3]

Data jsou údaje v databázi, které v databázi uchováváme a existují zcela nezávisle na programech. Mohou být také chápána jako známé fakty, které mohou být uloženy a mají svůj implicitní význam. Oproti tomu informace jsou data zpracovaná tak, že jsou srozumitelná. Tedy data se změní na informaci poté, co jsou zpracována. Jednoduše řečeno jsou data to, co ukládáme a informace to, co získáváme. Data lidé třídili a ukládali již dlouho před příchodem databází, ať již to bylo v archivech, knihovnách či diářích. Jde tedy o různé metody přístupu k datům, ať už ručně či přístup ve formě hromadného zpracování dat a již zmiňované databáze. [2]

Základem pro zpracovávání dat je souborový přístup, využívající systém řízení souborů,

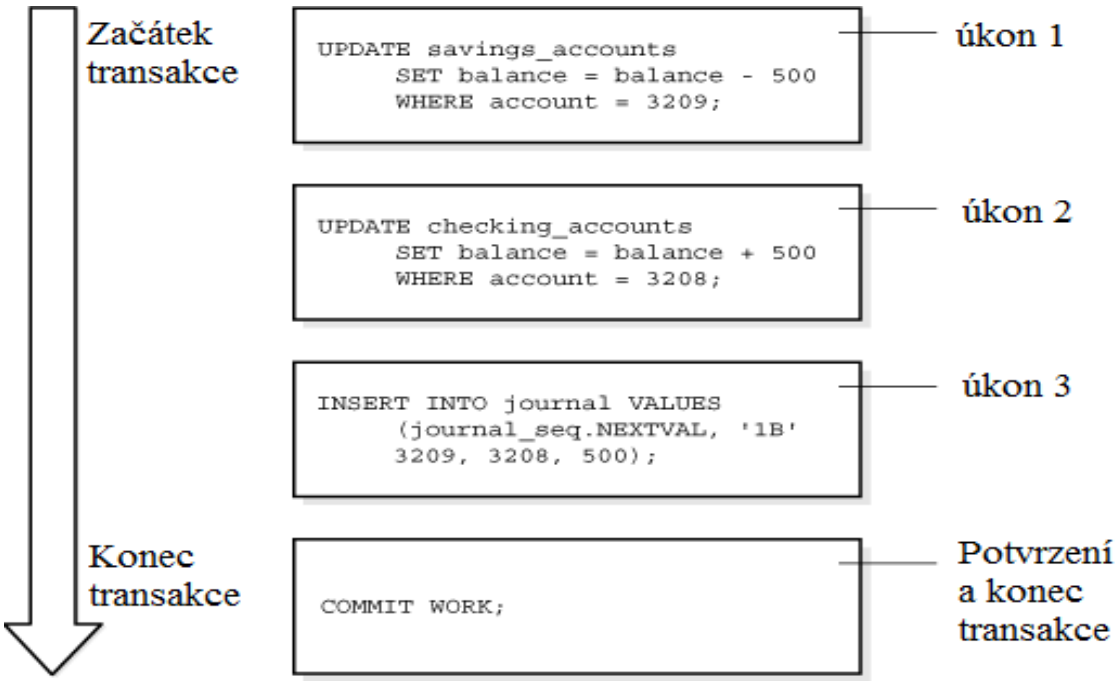
jenž poskytuje podporu pro různé ukládací metody a nabízí i možnost indexace, což umožňuje rychlý přístup k záznamům. Nevýhodou takto zpracovaných dat je jejich redundance a nekonzistence a již zmiňované problémy přístupu, izolace, ochrany a integrity. [1]

## 3.1 Databázová technologie

Databázová technologie obsahuje služby, jenž jsou poskytovány systémem řízení báze dat. Mezi těmito službami je zahrnuto transakční zpracovaní, řízení přístupu více uživatelů najednou, dále řízení zotavení se z chyb, řízení paměti a řízení ochrany dat, jisté jazykové prostředky a odolnost proti chybám a dnes také datové modelování, čili návrh databáze. Základem této technologie je oddělení systémových uživatelů a zajištění nezávislé komunikace s datovou základnou, respektive databází. [1, 4]

### 3.1.1 Transakční zpracování

Již zmiňovaný systém řízení báze dat je v převážné většině založen na principu transakčního zpracování. Transakcí rozumíme určitou posloupnost databázových operací, například může být transakce tvořena množinou příkazů v jazyku SQL pro manipulaci s daty, která je považována za logickou jednotku práce s databází. [1]




Obrázek 1: Struktura transakce
Zdroj: http://docs.oracle.com/database/121/CNCPT/transact.htm#CNCPT038

Základním principem transakčního zpracování je to, že se transakce buď provede celá, nebo se neprovede vůbec, což znamená, že pokud selže byť jen jeden jediný příkaz transakce, tak se do databáze nepromítne žádná změna provedená touto transakci. Databázovou operací jsou obecně míněny dva druhy operací a to buď operace přístupu k datům, či operace pro práci s daty. Zbylé operace jsou charakterizovány jako specifické.

Začátek transakce je specifikován operací START, operace COMMIT označuje její běžné ukončení a operace ABORT vyjadřuje nepřirozené ukončení a nutnost anulovat efekty, které byly transakcí vyvolány. Transakci je možné pojmenovat a skrze toto pojmenování ji vyvolat v pozdějších okamžicích, nebo také můžeme stejně jako v operačním systému vytvořit bod obnovy, na který je poté možno se za pomocí příkazu vrátit, což umožňuje návrat do vytvořeného místa a nemusíme tak anulovat veškeré efekty transakce, ale pouze jejich část.

Body obnovy představují uchování databáze v určitém stavu, je však potřeba volit je rozumně, zejména tak, aby při návratu do tohoto bodu byla databáze i nadále v konzistentním stavu. Problémem transakčního zpracování je rozvržení operací několika současně běžících transakcí najednou. Databázová technologie ovšem nabízí techniky zotavení se z chyb, jenž zabraňují nevhodnému ovlivnění právě ostatních běžících transakcí nebo samotné databáze. [1, 2, 3, 4]

### 3.1.2 Uživatelé databázového systému a ochrana dat

Ochrana dat představuje zabezpečení dat před neoprávněným zásahem před uživatelem či programem. Uživatelé mají svá specifická takzvaná uživatelská práva, která mohou být rozličná na různých úrovních databáze. Chráněn formou autentizace je také počáteční přístup do databáze. Autentizace je jistá forma přihlášení se do systému před započetím prací na databázi. Zmínka o rozličných uživatelských právech nutně vede k rozlišení uživatelských rolí v databázovém systému. Role jsou poté přidělovány uživatelům, což je přehlednější, než rozdělovat práva jednotlivým uživatelům, zejména proto, že většina uživatelů bude koncových a budou mít tuto roli předdefinovanou. Tito uživatelé se dají rozdělit do dvou skupin. [1, 3]

Jedněmi uživateli budou uživatelé nárazoví či příležitostní, kteří požadují data z databáze v různých souvislostech, své požadavky mění dle své potřeby a obvykle umí používat silnější dotazovací jazyk jako je například SQL. Druhou skupinou jsou uživatelé naivní, kteří pro přístup k databázi využívají aplikaci pro ně napsanou a zjišťují informace skrze aplikaci. Takovéto informace aplikace získá z databáze též formou dotazů, které ovšem již nejsou rozmanité a jsou v aplikaci definované a uživatel se tak skrze aplikaci nemůže dostat k jiným

informacím, než mu aplikace umožňuje. Ta je vytvářena aplikačními programátory, což je další z možných rolí v databázovém systému. Role správce dat není zpravidla chápána jako jeden člověk, nýbrž jako útvar, zabývající se vyhodnocováním přístupu do databáze, úpravou přístupových cest a také zabezpečuje autorizovaný přístup k databázi. [1]

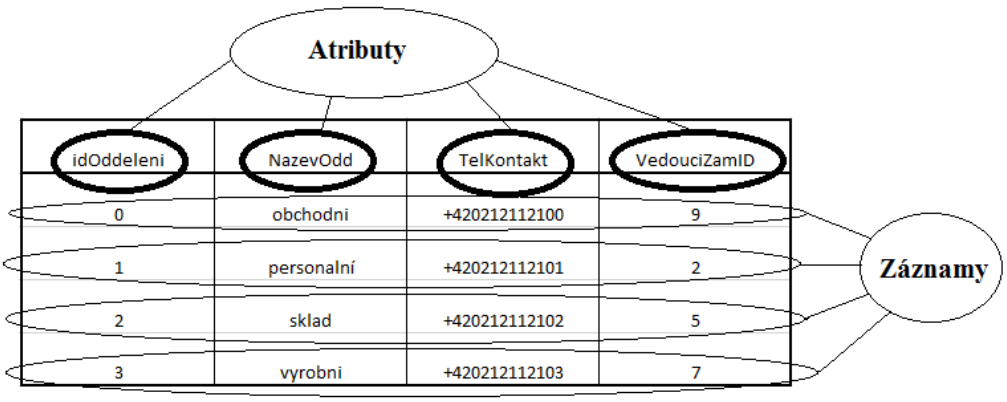
# 4 Relační databáze

Relační model databáze byl poprvé představen v roce 1970 doktorem Edgarem Frankem Coddem, také známým jako Ted Codd, jenž pracoval jako výzkumník u IBM. Doktor Codd svým modelem, který je založen na principu matematických relacích, zejména na teorii množin a na predikátové logice, která je nadstavbou výrokové logiky, vyvolal okamžitý zájem. Relační databáze jsou dnes hojně využívanou a neodmyslitelnou součástí všedního života, i když si to nemusíme vždy uvědomit, je jisté, že relační databázi využíváme prakticky dennodenně. Využitím relační úvahy vznikly základy dotazovacího jazyka SQL. [1]

## 4.1 Relační model dat

Relační databáze je spojována s matematickou úvahou zejména kvůli své podobnosti se strukturou zvanou relace. Relace je množina prvků, tvářících se jako řádek tabulky, přičemž relace představuje určitý vztah mezi tabulkami.

Relační model dat je jistou analogii tabulek, kdy navrhujeme relace při modelování, a proto hovoříme o relačním modelu dat, navrhování relací je tedy jeho nedílnou součástí. Tabulka je základním úložištěm v databázi a její data jsou uložena v řádcích a sloupcích. Uživatelé relačních databází vidí data jako tabulky. Relační tabulka je organizována tak, že řádky tabulky nejsou uloženy v určitém pořadí. Každá relace má své jméno a svůj atribut, který má též své jméno, svůj datový typ a svojí doménu, což je množina atomických hodnot. Každá hodnota v doméně je tedy dále nerozdělitelná. Při pohledu na tabulku relačního modelu dat, představují sloupce atributy a řádky tabulky pak jednotlivé záznamy viz obrázek. [1, 4, 5]




Obrázek 2: Struktura tabulky Zdroj: Vlastní tvorba

Návrh relační databáze vyžaduje schéma relací a schéma provázanosti tabulek neboli vztahů mezi nimi. Vztahy mezi jednotlivými tabulkami mohou být jedna ku jedné, jedna k n, nebo m k n, což je v praxi realizováno za pomoci vztahů jedna ku n, přes pomocnou tabulku. Návrh databáze je důležitá část tvorby databáze, neboť se od správného návrhu vše odvíjí a návrh by měl dodržovat normální formy nebo je poté třeba provést jeho normalizaci, což nedostatky odstraní. Výhodou relačního modelu je jeho vestavěná víceúrovňová integrita, také logická a fyzická nezávislost dat na databázové aplikaci. Relační model též obsahuje konzistentní a přesná data a spolu s jejich snadným získáváním se stal nejobvykleji využívaným databázovým modelem. [2]

Integrita v databázovém pojetí představuje splnění definovaných pravidel neboli splnění integritních omezení. Při zmínce o tom, že tabulka jako taková není v podstatě nijak organizována, je třeba uměle vytvořit evidenci záznamů tak, abychom byly jednoznačně schopni rozeznat, který záznam v tabulce je který. Tím se do jisté míry zabývá entitní integrita. Pro nás to znamená zavedení primárního klíče pro danou tabulku. [2]

Primární klíč je množina hodnot, které jednoznačně identifikují jednotlivé záznamy. Tímto klíčem může být nějaký unikátní atribut v naší tabulce, nebo námi uměle vytvořený atribut. Při výběru z již existujících atribut je třeba vhodně vybrat z možných kandidátů a zamyslet se nad opravdovou unikátností daného kandidáta pro danou tabulku. Součástí relačního modelu je schéma relační databáze, což je množina schémat relací a množina integritních omezení. [1]

Databázové schéma je logické uspořádání objektů databáze, jako jsou tabulky, pohledy, omezení a jiné součásti databáze. Schématem lze i omezit přístup jednotlivých uživatelů k jednotlivým objektům databáze. Pro práci s daty využívá model silného nástroje, jímž je relační algebra. [4, 6]

Relační algebra je jazyk, který pracuje s celými relacemi a ne jen s jejich n-ticemi a výsledkem operací jsou znovu relace. Základní operace tvoří kartézský součin, sjednocení, rozdíl, projekce a selekce. Selekce se vyznačuje za pomoci logické podmínky, jinak řečeno selekce je jistá forma podmínky. Analogii selekce z programování by bylo klasické podmínění formou if. Aplikací selekce na konkrétní relaci získáme určitý výsledek, který byl vymezen selekcí. Například záznamy z tabulky, které mají atribut větší než námi stanovená hodnota v podmínce. Projekce je jednoduše řečeno zúžený výběr. Projekcí vybereme pouze námi zvolené atributy z tabulky. Spojením selekce a projekce, představuje již zajímavější dotazování nad naší tabulkou respektive databází. Neméně potřebnou operací při dotazování se nad databází je operace spojení. To představuje spojení dvou tabulek respektive sjednocení jejich atributů a opět je možno spojení kombinovat s operacemi projekce a selekce, což umožňuje tvorbu více tabulkových dotazů. [1]

## 4.2 Dotazovací jazyk

Základ relačního datového modelu umožnil vznik dotazovacího jazyka SQL. Jazyk je množina řetězců nad abecedou, kdy řetězce jsou tvořeny na základě pravidel dané gramatikou, a gramatika určuje syntaxi výrazů jazyka. Dotaz představuje nějakou funkci, jenž je definována nad prostorem všech přípustných databází s daným schématem. Výsledkem bude znovu databáze s určitým schématem. Dotazovací jazyk musí být určitou formou provázán s databází, tedy formulované dotazy musí být nad databází vyhodnotitelné. Pro funkci jako dotaz tedy musíme být schopni vytvořit algoritmus, jehož výsledkem musí být hodnoty z naší databáze, nepoužíváme-li výpočetní či jiné funkce, které vytvářejí z našich dat nové informace. Výsledek není závislý na reprezentaci databáze. [1, 3]

Oproti relačnímu principu, SQL nemluví o relacích nýbrž o tabulkách a umožňuje do nich zavést i prázdné hodnoty. [1] Null, neboli prázdná hodnota zastupuje chybějící či neznámou hodnotu a jako taková je hodnota null užitečná, pokud je při návrhu třeba mít údaje nepovinné či údaje, které v danou chvíli není možné vyplnit. Hodnoty null však mají nevhodný dopad na matematické operace, které nám tak dávají nepřesný či žádný výsledek. [2]

Dotazovací jazyk by měl být omezený a expresivní. Splňuje-li obě tyto vlastnosti, nazýváme takovýto jazyk, jazykem úplným. Expresivní jazyk je jazyk, který nám umožňuje vyjádřit jakýkoliv databázový dotaz. Vyjadřovací silou databázového jazyka je množina všech dotazů, jenž můžeme vyjádřit. Dotazovací jazyky obecně umožňují ptát se jak na data, tak i na metadata a jejich principy lze uplatnit i v dalších příkazech pro manipulaci s daty. Možnosti dotazování v SQL jsou široké a SQL nabízí i různé agregační funkce a aritmetiku. [1, 5]

### 4.2.1 SQL

Strukturovaný dotazovací jazyk, zkráceně SQL, jehož kořeny sahají do roku 1974, je jazykem relačních databází společnosti IBM. Skutečný vývoj a rozsáhlá implementace SQL započala v 80. letech minulého století s nástupem počítačů. Vývojem od sytému řízení báze dat s SQL pro jednoho uživatele, přes databázové servery založené přímo na technologii SQL a zlom v podobě standardizace organizací ANSI přivedl databázové modelování v SQL do dnešní podoby. [1]

Charakteristickým rysem je ukládání dat v databázi formou tabulek, ať již skutečných či virtuálních ve formě pohledů. Pohled je tedy virtuální tabulka obsahující pole jedné nebo víc tabulek v databázi, fyzicky však data neobsahuje, obsahuje pouze předpis, jak mají být data získána. Využíváme je k výpisům souvisejících informací s využitím návaznosti dat na

sebe. Hlavní výhodou pohledů je spojení dat z více tabulek a práce s nimi, navíc pohledy jsou pouze ke čtení a tudíž zabraňují uživatelům pracovat s určitými daty. Pohledy je možno též využít pro implementaci integrity dat. [2]

Dalším rysem SQL je to, že vrací data programu, který se dále nemusí starat ani o strukturu, ani o umístění dat. Poloha tabulek v databázi ani pořadí sloupců a řádků v tabulkách není pro SQL důležité, neboť tabulky a sloupce jsou identifikovány jménem a řádky jsou identifikovány svými hodnotami ve sloupcích. SQL nabízí možnost indexace, což usnadňuje přístup k řádkům a proces přístupu je tak rychlejší. [6]

Součástí jazyka SQL je jazyk pro definici dat, jazyk pro manipulaci s daty, též jazyk pro manipulaci s daty v hostitelské verzi a dynamický SQL, dále jazyk pro definici přístupových práv, jazyk modulů a řízení transakcí. Podpora tvorby, modifikace a odstranění definic tabulek a pohledů je obsahem jazyka pro definici dat, k němuž je možné připojovat i definice integritních omezení. Formulovat dotazy a vkládat, odstraňovat a modifikovat řádky tabulky umožňuje uživateli, jak již z kontextu vyplývá jazyk pro manipulaci s daty, který je do jisté míry ovlivněn relační algebrou. Konstruovat dotaz v průběhu programu je umožněno dynamickým SQL, jenž umožňuje tvorbu dotazů při běhu programu. Jazyk modulů nabízí možnost oddělených SQL příkazů od programu a jejich následného připojení. Provedení transakci je možno řídit skrze několik zabudovaných SQL příkazů v části řízení transakcí. SQL dále nabízí při tvorbě tabulky možnost globální či lokální dočasné tabulky, což umožňuje zachování struktury tabulky pro zobrazování mezivýsledků, které není třeba ukládat. Data v dočasných tabulkách nezůstávají. SQL podporuje a rozlišuje přesné a aproximativní numerické typy, znakové řetězce a časové datové typy. Zkrátka datové typy, které známe z klasických programovacích jazyků a ještě nějaké další navíc, které bychom museli v programovacích jazycích ručně dotvářet jako makra či šablony. [1, 5, 6]

Pro manipulaci s daty se používají příkazy select, insert, delete a update, přičemž největší složitost z logického pohledu je v příkazu select, jelikož je potřeba zvolit tabulku, ze které budou data vybrána, což vyžaduje jistou znalost schématu databáze. Pro definici dat jsou to příkazy create, alter a drop, jenž jsou využívány zejména při tvorbě, úpravě a mazání tabulek. Dále SQL nabízí příkazy k řízení přístupových práv grant a revoke, které spolu s definičními příkazy umožňují i tvorbu rolí v databázi. Též jsou zde příkazy pro řízení transakci start transaction, commit a rollback. [1, 6]

Produkty z databázového prostředí používají SQL a spojení SQL a nejznámější společnosti Oracle nabízí skvělou a snadnou možnost jak pracovat s databázemi. Připočteme-li možnost využití funkcí, jež SQL nabízí je možnost dotazování opravdu obrovská a využijeme-li ještě vnořené dotazy a budeme-li pracovat s různými mezivýsledky, umožňuje nám SQL ptát se

prakticky na cokoliv co je potřeba v souvislosti s námi navrženou databází, popřípadě po prodiskutování s požadavky zákazníka a zvážením reálnosti a užitečnosti patřičného dotazu. SQL má tedy velmi silnou vyjadřovací sílu, která je podpořena vestavěnými agregačními funkcemi.

# 5 Integrita dat

V relačním modelu dat může být integrity dosaženo za pomoci pravidel či integritních omezení. Integritní pravidla jsou zabudována již ve schématu databáze, jsou obecná a vše musí těmto pravidlům odpovídat. Omezení mohou být definována i za běhu a databázový systém se pak těmito omezeními řídí, a pokud uživatel způsobí něco, co odporuje omezením, systém tuto činnost zastaví. [6] Samotný pojem integrita lze laicky chápat jako neporušenost, nedotknutelnost a celistvost, v našem případě dat v databázi. V databázových systémech je korektnost a přesnost dat velmi důležitá a jednou z možností jak data v této podobě zachovat je právě kontrola integrity. Ta se stará o to, aby uživatel svou neznalostí či nepozorností nenapáchal v databázi chyby. [7]

Integrita může být řešena ve fyzické a logické vrstvě, přičemž fyzickou integritou myslíme integritu ve spojení s hardwarem. Tato práce se bude věnovat logické formě integrity. Integritní omezení jsou určitými průvodci pro uchování dat v korektní formě. Specifikuje též co je s daty možno provádět. Veškerá pravidla a omezení jsou neustále aplikována na všechna data vstupující do databázového systému. To vše zajišťuje, že výsledkem je databáze v korektním stavu. [8]

Hlavními výhodami integritních omezení je možnost definice omezení za použití SQL příkazů, což nevyžaduje žádné další programy, které by byly třeba. Další výhodou, kterou integritní omezení nabízejí, je jejich definice pro tabulky a to že jsou uložena v databázi samotné. Omezení lze též dočasně vypnout za účelem prevence systémové náročnosti při nahrávání velkých množství dat. Výhoda využití integritních omezení sebou však přináší určitou ztrátu výkonu. Obecně platí, že integritní omezení odpovídá zhruba svou náročností jako provedení SQL dotazu, který dané omezení vyhodnocuje. Při veliké náročnosti omezení a slabším hardwaru, je tedy možné postřehnout určitou ztrátu výkonu a prodlevu v provedení potřebné akce.[15]

Datovou integritu lze garantovat použitím triggerů, využitím zabudovaných procedur, kódem v databázové aplikaci, či již zmiňovanými integritními omezeními. Využitím validačního pohledu při implementaci integrity získáváme širší možnost kontroly. Účel validačního pohledu je stejný jako účel validační tabulky. Jde o zobrazení respektive uchování skupiny dat, které jsou důležité pro datovou integritu, to znamená souhrn většího množství dat, na kterých jsou aplikována omezení a pravidla na kterých nám záleží.[14] Rozdíl mezi validační tabulkou a validačním pohledem je ve způsobu tvorby, tabulka ukládá validační data, validační pohled je většinou získává pouze z jedné tabulky, na kterou se implementují jednotlivá omezení.

Integritu dat respektive integritní omezení rozdělujeme do tří kategorii. Jde o integritu entitní,

integritu referenční a integritu doménovou. Integritní omezení mohou fungovat i rekurzivně a to ve smyslu omezení hodnoty v tabulce, která odkazuje sama na sebe. Zabráníme například tomu, aby člověku, jenž vede oddělení, šéfoval jeho zaměstnanec. Aplikovat můžeme i dynamická omezení, jako je omezení že původní hodnota nesmí být větší než hodnota nová. Jednotlivá omezení můžeme vyjádřit za pomocí relační algebry, nebo za pomocí relačního kalkulu, či dnes nejčastěji za pomoci SQL. Využitím integritních omezení se vyhneme problémům jako je nekonzistence dat a můžeme zamezit i nevhodným či nežádoucím vazbám mezi jednotlivými daty. [9]

## 5.1 Entitní integrita

Entitní integrita je ať již z logického pohledu člověka tak i z relačního modelu vyžadována. Smyslem je zavedení primárního klíče pro každou tabulku. Primární klíč je tak nedílnou složkou tabulky, což nám pomáhá i při identifikaci jednotlivých řádků respektive záznamů v databázi. Entitní integritní omezení tedy jasně stanovuje, že každá tabulka musí mít primární klíč a ten nesmí být null, neboť prázdný klíč by znemožňoval identifikaci záznamů v tabulce a neumožňoval by pak využití následných možných referenci. [4]

Entitou je objekt našeho zájmu. Fyzicky je můžeme chápat jako označení skupiny, či jednotlivce. Například zaměstnanci, objednávky, odběratelé a tak dále. Na základě vztahů mezi entitami vzniká právě potřeba entitní integrity, která je realizována primárními klíči a správné určení entit v naší databázi je neméně důležité. [13]

Primární klíč se skládá z jednoho nebo více atributů, což vytváří unikátní údaj, tedy takový údaj, který nebude shodný s žádným jiným údajem v dané tabulce. Primární klíče tedy představují unikátní identifikační funkce v relačním modelu dat.[6] Například firma Oracle respektive její databázový software vyžaduje omezení indexace u primárního klíče. Primární klíč nikdy za žádných okolností nesmí mít hodnotu null a měl by být minimální a to ve smyslu, že pokud je tvořen více atributy, nelze žádný atribut odebrat, aniž by se ztratila jedinečnost primárního klíče.[11] Jako primární klíč můžeme volit již existující atribut či atributy nebo zvolíme generování primárního klíče databázovým systémem, takzvaný umělý klíč, což přidá nový atribut, zpravidla číselný. Nejjednodušším primárním klíčem jaký si můžeme představit je rodné číslo při evidenci lidí, či číslo občanského průkazu, tedy unikátní údaj, který je spárován pouze s daným člověkem.

## 5.2 Referenční integrita

Referenční integrita má co do činění s vazbami respektive vztahy mezi tabulkami. Jednotlivé vztahy v databázi jsou vyjádřeny schématem a realizace těchto vztahů je v tabulkách provedena odkazem na jinou tabulku. Odkazem je v těchto případech takzvaný cizí klíč, který ukazuje na primární klíč jiné tabulky. Takto je pak i zřetelná propojenost jednotlivých tabulek, odpovídající schématu databáze.

Cizí klíč je atribut či skupina atributů, vytvářející spojení mezi daty ve dvou tabulkách. Jinými slovy tvoří přímé spojení mezi dvěma tabulkami, takzvaný křížový odkaz. [10] Cizí klíč odkazuje na primární klíč jiné tabulky, čímž je dáno jasné propojení. Referenční integrita může být tedy specifikována jako propojení dvou relací respektive tabulek, zajišťující konzistentnost tohoto vztahu. [4] Pro cizí klíč musí platit určitá pravidla, která jsou realizována právě referenčními integritními omezeními. Vztah mezi cizím klíčem a primárním klíčem musí být smysluplný a musí odkazovat na nějaký existující primární klíč nebo hodnota v cizím klíči musí být null, čili danému záznamu není přiřazena žádná reference na záznam nebo záznamy další. Dále platí, že atribut cizího klíče má stejnou doménu jako atribut primárního klíče, na který se odkazuje.

Referenční integrita je nedílnou součástí tvorby databáze a její omezení jsou tvořeny na základě relačního schématu dané databáze. Jakákoliv změna primárního klíče musí být aplikována na všechny cizí klíče, nebo nemůže být aplikována vůbec. Vyplývá to z nutnosti zachování konzistence. Pokud bychom změnili primární klíč a nezměnil by se klíč cizí, vztah, který by byl mezi těmito klíči, by již nebyl funkční, jelikož by nastala situace, kdy by cizí klíč odkazoval na neexistující klíč primární což je v rozporu s referenční integritou a integritní omezení by tak bylo narušeno a databázový systém by hlásil chybu, jelikož je tato operace nemožná. Správné integritní omezení by tedy změnilo i všechny cizí klíče, nebo by zneplatnilo veškeré vztahy s původním primárním klíčem.

Referenční integrita může specifikovat určité akce, které budou provedeny na určitých řádcích v tabulce, která je potomkem v případě, že je hodnota referenčního klíče měněna nebo mazána. Dojde-li na takovouto činnost, je možné integritnímu omezení říci, jak se má v takové situaci zachovat. Jedná se o akce update, delete no action a delete cascade, eventuálně restrict a nullify. Klíčovým slovem restrict uživateli zamezíme provádět změnu v referenci. Nullify ovšem nastaví hodnoty, které se vztahují na akci s tímto spojenou, na null. [12] Jde o případy, kdy se smaže záznam, na který odkazují nějaká jiná data a ty místo odkazu na neexistující záznam mají hodnotu null. Update je umožněn, pouze pokud všechny řádky budou i nadále mít validní referenční klíč. Delete no action neumožňuje aktualizaci či smazání referenčního klíče, pokud by výsledná data narušila pravidla referenční integrity. Nemůžeme

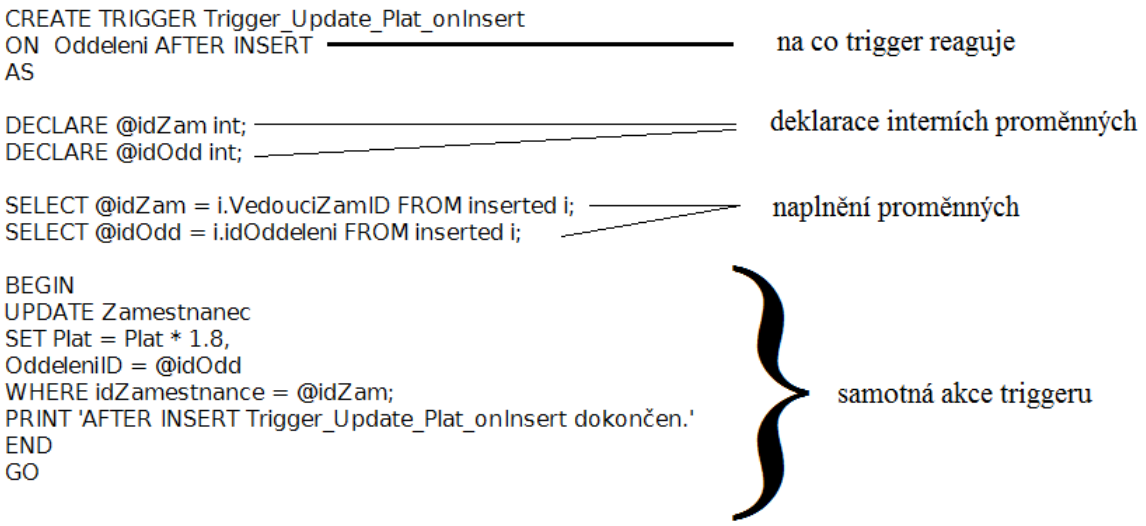
tedy smazat údaje, na které jsou vázána nějaká jiná data respektive záznamy. Delete cascade, neboli smazat kaskádově znamená, že bude vybraný záznam odstraněn a spolu s ním i všechny záznamy, které se na něj odkazovali a v případě změny dat, dojde i ke změně záznamů na nově změněnou hodnotu. Akce tohoto typu dopadne vždy v pořádku. [15]

## 5.3 Doménová integrita

Doménová integrita je integrita týkající se dat samotných, tedy našich záznamů v databázi. To znamená, že požadujeme zachování korektního významu dat a jednotlivé atributy patří do nějaké domény a mají svůj název a datový typ. Někdy je též nazývána integritou na úrovní polí a umožňuje zachování struktury jednotlivých polí, čili hodnoty jsou konzistentní, platné a přesné a jednotlivé atributy stejného typu jsou v celé databázi definovány stejně, tedy konzistentně.[2] Doménové integritní omezení jednoduše představuje omezení konkrétní hodnoty atributu. Může se jednat o omezení například rozsahu čísel, které je možno zadávat, či omezení přesného tvaru, ve kterém mají být hodnoty zadány, což se dá chápat a využít jako určitá šablona.[6]

Relační model při správném návrhu databáze defakto vynucuje určité prvky doménové integrity, respektive využívá určitá omezení. Tato omezení se dají při znalosti SQL vcelku jednoduše aplikovat a relativně snadno definovat. Trvání doménových integritních omezení nemusí nutně být nekonečné, omezení je možno přidávat, mazat, měnit či dočasně nevyužívat některé definované omezení. [16] Kontrola omezení může býti odložena a to na konec transakce, která se právě provádí. Pokud je nějaké omezení porušeno, transakce je stornována. V případě, že neprovádíme odloženou kontrolu, se kontrola integritních omezení provádí po každém příkazu. [15] Složitější omezení mohou vyžadovat řešení triggery.

Trigger je procedura, která je zaktivována na základě nějaké události, jako je vložení, úprava či smazání záznamu v tabulce. Trigger smí být definován pouze pro tabulky a může obsahovat SQL příkazy, které budou provedeny, nebo může vyvolat nějakou jinou uloženou proceduru databáze. Na rozdíl od uložených procedur databáze, trigger je spuštěn bez ohledu na to jaký uživatel či aplikace s daty respektive s databází pracuje. [16]




Obrázek 3: Struktura triggeru Zdroj: Vlastní tvorba

## 5.4 Integritní omezení v SQL

První standard SQL umožňoval využít pouze omezení not null a unique. Omezení not null zaručuje danému atributu to, že vždy bude mít nějakou hodnotu. Unique garantuje jedinečnost hodnoty, tedy žádná hodnota v daném sloupci v dané tabulce se neopakuje. Pro kontrolu nějaké hodnoty se používá klíčové slovo check. Klíčové slovo je slovo nebo indikátor, mající specifický význam v daném programovacím jazyce. Check je námi definované pravidlo a používá se tedy pro kontrolu hodnoty, například rozsahu integer hodnoty, což v případě neshody vyvolá chybovou hlášku, nebo eventuálně můžeme sami říci co a jak má systém v takové situaci udělat, nebo jak má jednat. SQL umožňuje při deklaraci tabulky nastavit i defaultní, tedy implicitní hodnotu, což je hodnota, která v případě nevyplnění dané položky vkládaného záznamu bude nastavena na příslušnou hodnotu. [1]

Referenční integrita se obstarává při deklaraci explicitního primárního klíče klíčovým slovem primary key, které v sobě vlastně obsahuje omezení not null a unique. Obdobně lze přidělit atributu klíčové slovo foreign key a deklarovat tak atribut jako cizí klíč a nastavit mu tak příslušnou referenci na klíč primární. SQL nabízí též širší možnosti pro pravidla referenční integrity jako zamezení smazání, či upravování dat, na které je odkazováno. Také lze nastavit, aby data, která se odkazovala na nějaká jiná, byla po změně či smazání odkazovaných dat, nahrazena hodnotou null nebo implicitní hodnotu. Též lze využít kaskádovité možnosti k změně dat, jenž jsou odkázána na data, která byla změněna. Jednotlivá omezení se mohou nacházet v rozličných stavech. Omezení může být povoleno, což zaručuje kontrolu

dat s daným omezením a jejich správnost, včetně ověření dat nově vstupujících. Opakem je vypnuté, tedy nepovolené omezeni. Validní omezení garantuje, že existující data jsou v pořádku, přičemž splňují daná integritní omezení, která se na ně vztahují a stav bez validace nezaručuje správnost již existujících dat. [15]

Po vytvoření jakéhokoliv integritního omezení je třeba provést jeho verifikaci. Kontrolujeme správnost syntaxe, zda-li je well formed, respektive zda-li je z logického hlediska v pořádku. Též kontrolujeme, zda-li toto omezení nenarušuje stav databáze a zda-li již není zahrnuto v nějakém jiném omezení, tedy zda-li není redundantní. Zavedená integritní omezení by neměla odporovat sami sobě a měla by být konzistentní. [7]

Aplikace integritních omezení může být náročná, a proto je třeba omezení optimalizovat. Množství kontrolovaných dat může být zredukováno prováděním omezení pouze na relevantních částech relací. Omezení můžeme vyjádřit i algebraicky, pokud je to možné a tak mohou být snadněji vyhodnocena. Pokud je databáze rozkouskovaná, můžeme této znalosti využít a použít co nejméně fragmentů při aplikaci potřebného omezení, též využíváme znalosti databáze a prostředí, pro které je tvořena. [7]

### 5.4.1 Nastavení omezení

Jednotlivým omezením můžeme nastavit, zda-li kontrola těchto omezení má být aplikována ihned po provedení příkazu, nebo až na konci provedené transakce a to dle standardů ANSI SQL92. Nastavení omezení může trvat do konce průběhu transakce, nebo do další změny nastavení daného omezení. Pokud vyhodnocujeme omezení okamžitě po každém příkazu v dané transakci, databázový systém nejprve kontroluje odložena omezení a pak okamžitě navazuje okamžitá kontrola omezení, které mají nastaveno, aby se kontrolovaly vždy po příkazu. Pokud některý příkaz nesplní některé omezení, operace rollback navrátí systém do stavu před provedením transakce, nebo je možné se pomocí tohoto příkazu navrátit do určitého stavu databáze, nebo do záchytného bodu, který byl například vytvořen před započetím transakce. Operací commit potvrdíme změny provedené v databázi. SQL příkaz alter umožňuje nastavit všem omezením, aby byla provedena okamžitě, či až na konci transakce. Okamžitá kontrola omezení je cestou, která nám umožní včasné rozpoznání případné chyby a zabrání vracení celé sady příkazu. [15]

## 5.5 Business pravidla

Poté co je struktura tabulek a polí dle pravidel správná a vkládaná data jsou rovněž v pořádku a nacházejí se v konzistentním stavu a vztahy v databázi jsou smysluplné, je třeba navrhnout

určitá business pravidla. Jedná se o pravidla formulující určité omezení na konkrétní část databáze. Jejich formulace závisí na získávání a zpracování dat dané organizace, pro kterou databázi tvoříme, též se snažíme uchovávat pouze data, která budou běžně využitelná a ne jen pouze potencionálně. [2]

Business pravidla lze rozdělit na dva typy a to na databázově orientovaná a aplikačně orientovaná pravidla a také je lze dělit dle kategorií na pravidla specifická pro pole a pravidla specifická pro vztahy. Databázově orientovaná pravidla lze zavést při návrhu databáze, například pozměněním vlastností polí, nebo vhodně upravíme vztahy mezi tabulkami. Zdánlivě jasně logická úvaha při návrhu databáze pro danou firmu nad poli či vztahy dat ve firmě je vlastně právě implementace business pravidel. [2]

Aplikačně orientovaná pravidla budou zavedena v rámci aplikace, která nad naší databází pracuje. Jde o pravidla, jenž nelze či nemá smysl aplikovat při logickém návrhu databáze. Specifické vlastnosti a formát polí jsou analogií k doménové integritě. Chceme mít v databázi určitý formát pole, nebo jiné pole smí obsahovat pouze určité symboly či výčet symbolů, přičemž nejsnadnější implementace pro výčet symbolů či zkratek je kromě relační algebry využití validační tabulky. Zbytečně složitě by se aplikovalo omezení pro konkrétní atribut, pokud bychom chtěli, aby odpovídal výčtu hodnot, které jsou přípustné. K tomu velice jednoduše poslouží validační tabulka nebo pohled, jenž bude obsahovat výčet povolených údajů a jednoduchým SQL dotazem pak budeme toto omezení aplikovat. [2, 14]

Business pravidla pro vztahy upravují vztahy mezi tabulkami. Návrh takového pravidla se odvíjí od logické struktury a reálných možností firmy a jsou ve své podstatě součástí každého návrhu, i když si to třeba plně neuvědomujeme. Business pravidla dotvářejí celkovou integritu dat v databázi a na základě těchto pravidel tvoříme omezení, která jsou vhodná pro konkrétní organizaci či firmu.

# 6 Momentální situace problematiky

V dnešní době je datová integrita stále podstatnou a měla by být i nedílnou součástí každé databáze. Nové i staré podniky budují databáze, ve kterých je potřeba integritu zabezpečit a předejít tak možným problém. Aktuálním problémem je například projekt registr vozidel, který ukazuje, co vše se může stát, pokud není využit potenciál integritních omezení a vše je řešeno skrze nadstavbovou aplikaci a databázi využíváte pouze jako strukturované úložiště dat a údajů potřebných pro danou aplikaci. Problémovým byl i systém pro vyplácení sociálních dávek, který hrubým nedopatřením umožňoval opakované vyzvednutí sociální dávky, která již byla v daný měsíc vyzvednuta, ale systém z jakýchsi nepochopitelných důvodů tuto situaci nezaregistroval. S vývojem jazyka SQL se během čtyřiceti let vylepšování jazyk dostal až do dnešní podoby a je stále hojně využíván. Stejně jako normy ISO/IEC i norma ANSI má své oblasti, na které se soustřeďuje. ANSI pracuje na potřebě ekonomických služeb a snaží se podpořit rozvoj komplexních strojů spolu s větší ochranou spotřebitelů a pracovníků.[17] Jazyk SQL během let umožnil i práci s xml dokumenty a dnes umí pracovat i s javascriptovými zápisy. Společnost Oracle v roce 2014 vydala projekt nazvaný Big data SQL, což je software, který umožňuje bezpečně a bez problémů integrovat velká data, ať již skrze framework Hadoop, či klasicky přes SQL, nebo využitím moderního NoSQL.

Software Big data SQL běží na platformě big data appliance, přičemž kombinace této platformy a softwaru big data vede k implementaci SQL spolu s velkoobjemovými daty. Využitím SQL dotazů a analýzy dat napříč celým systémem, již není potřeba kopírovat a přesouvat data mezi různými platformami. Oracle s tímto přístupem přinesl řadu nových technologií a dovedností k zajištění zjednodušeného přístupu. Zjednodušil se zejména přístup k velkým datům a to kombinací relačních a nerelačních technologii což spolu s rozšířením SQL napříč celým systémem umožňuje snadnou analýzu dat. V souhrnu jde tedy o to, že software Big data SQL umožňuje napojit existující ochrany a mechanismy na framework hadoop a na NoSQL data. [18]

NoSQL je nový databázový koncept, který umožňuje horizontální i vertikální uložení dat. To je například uložení záznamu na různých uzlech. Tento databázový koncept popisuje datový model spíše intuitivně a ve stejném duchu je chápána i myšlenka atomicity, konzistence, izolace a trvalosti. Koncept NoSQL je tzv. schema free, čili schéma takové databáze je značně volně vyobrazené. Integrita dat je v tomto přístupu řeší spíše na bázi aplikace, která nad databází běží. Známé NoSQL systémy jsou Cassandra, kterou využívá například Facebook a Twitter, či Hadoop, který používá Amazon a Yahoo. [19]

V souvislosti se stále se rozšiřujícím službami na internetu je integrita obrovského množství dat klíčová zejména pro vyhledávání. Správně strukturovaná, korektní a přesná data

též usnadňují práci mobilním zařízením, které nedisponují až tak velkou výpočetní kapacitu a dále se pak snažíme o co možná nejmenší přenos dat, protože přenosová rychlost není taková, jaká je na počítačových sítích. Z ekonomického hlediska nám příprava a zabezpečení dat přinese i uspokojivou návratnost investic. [20]

Pro potřeby dnešní doby je tedy integrita dat naprosto nutná, zejména proto, že 30 – 50% chyb a narušení bezpečnosti je způsobeno lidmi, kteří s daty, databází, či systémem pracovali a svojí chybou tak narušení zavinili, i když se tak mohlo stát nedbalostí, nedopatřením, či omylem. Omezený přístup a kryptování dat nejsou stoprocentní a přidáním integritních omezení se rozhodně nic nepokazí. Proto je tedy třeba dávat si pozor, protože únik, nebo nekorektnost dat, může mít fatální následky pro podnik, neboť data a informace z nich získané jsou velmi cennou věcí. [20]

Jedním z aktuálních problémů v České republice je centrální registr vozidel. Tento problém řeší ministerstvo dopravy již od roku 2012 a během této doby bylo řešení několikrát přepracováno a ani dnes není v optimálním stavu. Hlavním problémem jsou často nekonzistentní data, které jsou v databázi obsažena. Opakovaná nerozhodnost při realizaci centralizovaného, nebo necentralizovaného systému, též zapříčinila ohromné kolapsy a nedostupnost dat. Při jednom z přepracování projektu, bylo třeba přepsat data vybraná z původní databáze do nově navržené databáze, což odhalilo naprostou nekonzistenci dat v původní databázi. Ať již šlo o nemístné formátování dat, či data kompletně chybná, neúplná a tak dále. Integrita dat zde evidentně byla opomenuta, jinak by k takovýmto havarijním stavům nemohlo dojít.

Ani nový software se chybám nevyvaroval a evidoval majitele vozů, kteří již dávno byly po smrti. Kvalitním a pečlivým zpracováním a zapojením integrity by se projekt určitě vyvaroval mnoha nepříznivým stavům, ve kterých se nacházel. I přes zcela jasnou účinnost a platnost integritních omezení, je datová integrita stále nedoceněným řešením, které je v mnoha případech lepším řešením, než jakákoliv jiná.

# 7 Praktická část

## 7.1 Charakteristika modelové podnikové databáze

V rámci ukázky využití integritních omezení analyzuji výrobní podnik, živící se výrobou drobných elektronických zařízení, které jsou využívána převážně dalšími podniky pro kontrolu jejich výroby (ampérmetry, voltmetry atd.) či zapojení kontrolních elektronických zařízení do procesu výroby nebo provozu (alarmy, regulátory průtoku vody, různé ukazatele atd.). Příležitostný prodej i fyzickým osobám na základě objednávky. Jedná se o malý podnik, který nevynakládá žádné finanční prostředky na reklamu. Obsahuje ovšem několik oddělení, do kterých jsou zařazeni zaměstnanci. Dále se evidují výrobky, informace o odběratelích a objednávkách, které firma přijímá.

Z databázového hlediska řešíme interní evidenci zaměstnanců, oddělení, výrobků, objednávek, odběratelů a také evidenci služebních cest. Veškeré zajištění interních pravidel bude realizováno primárně skrze integritní omezení, což povede ke konzistenci a přesnosti dat. Předejde se tak nepřehlednosti, nepřesnosti a nekonzistentnosti. Dále se požaduje realizace povýšení zaměstnance na vedoucího oddělení a okamžité odečtení objednaných kusů výrobků na objednávce od reálného počtu kusů skladem, což úzce souvisí i s pravidlem nemožnosti objednat kusy, které skladem nejsou. To bude realizováno za pomocí triggerů. Je třeba správně vyřešit jednotlivá propojení tabulek a zaměstnanců s odděleními. Zaměstnanec nemůže například vést více než jedno oddělení na základě interních směrnic podniku. Objednávku vyřizuje zaměstnanec příslušného oddělení. Služební cesta je též logicky omezena a to tak, že žádný zaměstnanec nemůže být na dvou či více služebních cestách současně a má naplánovanou pouze jednu služební cestu maximálně na jeden den. Odběratel může objednat jeden a více výrobků, přičemž odběratelem může být fyzická i právnická osoba. Tyto pravidla budou realizována integritními omezeními.

Toto řešení je zkrácené pro demonstraci aplikace integritních omezení. V reálném řešením by databáze byla mnohem komplexnější a vyžadovala by zapojení dalších prvků z SQL praxe jako je zabezpečení přístupu uživatelů, rozdělení jejich rolí atd.

## 7.2 Realizace

Praktická část této práce byla řešena na SQL serveru 2008 Release 2, nainstalovaném na lokálním počítači, skrze SQL skripty zadávané přes SQL server management studio. Postup realizace databáze, přes počáteční návrh schématu, atributů a potřebných omezení, je

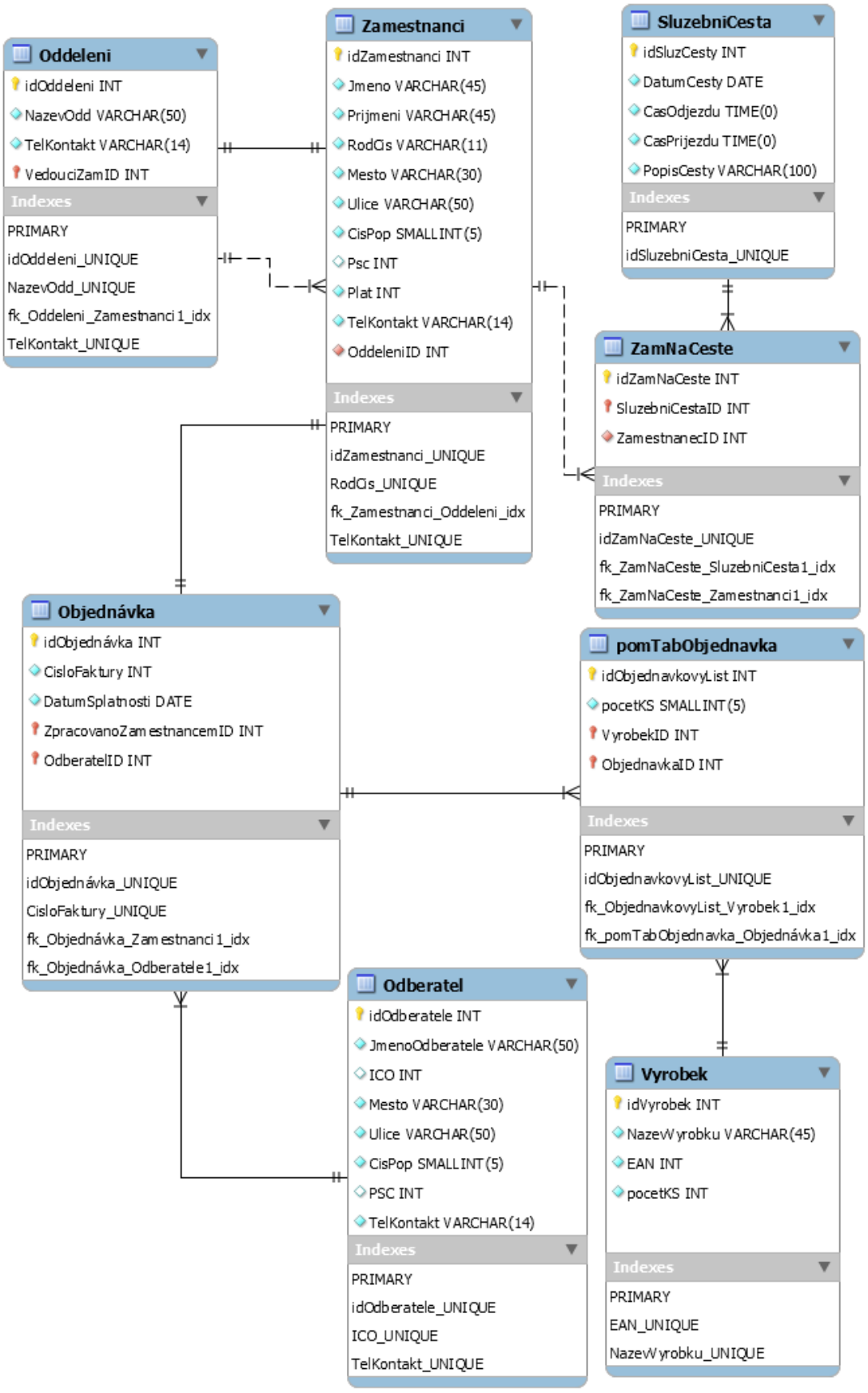
od tabulek, které sami neobsahují cizí klíč, tedy od jednodušších po složitější, kde je třeba zapracovat do řešení SQL funkce, které využívají jednotlivá omezení a návaznost tabulek, tedy reference skrze cizí klíče, která se mnohdy musí dodávat postupně. Tímto docílíme konzistentního návrhu databáze, na který pak může navázat informační systém či nějaká další aplikace, která bude využívat vytvořenou strukturu a bude s ní pracovat.

V našem řešení je často použita klauzule UNIQUE, což je nejsnadnější doménové integritní omezení pro stanovení jedinečnosti daného atributu, či skupiny atributů. Entitní integritu řeší klauzule PRIMARY KEY a v některých případech je vymezena i dalšími omezeními.

Klauzule NOT NULL vyjadřuje v celém zadání nutnost vyplnění dané položky. Stanovení klauzule IDENTITY u atributu značí jeho automatickou inkrementaci. Toto řešení je využito u všech primárních klíčů, což je vhodné řešení vzhledem k volbě klíčů umělých.

### 7.2.1 ER diagram pro interní návrh databáze

Diagram byl vypracován v programu MySQL Workbench a znázorňuje logické propojení tabulek vyjádřením jednotlivých vazeb mezi nimi. Vazba M:N je řešena pomocnou tabulkou a využívá tak vazby 1:N. Vazby propojení cizích klíčů s primárními klíči jednotlivých tabulek odpovídají standardům tvorby databáze a samy o sobě představují základní integritní omezení. Jednotlivé propojení pak z logického hlediska tvoří business pravidla, která byla požadována podnikem a v řešení byla zohledněna.




Obrázek 4: ER diagram řešení Zdroj: Vlastní tvorba

### 7.2.2 Create skripty

Veškeré skripty byly psány ručně, bez použití generátorů kódu.

**Tabulka Výrobek**

Pro evidenci výrobků podniku je třeba uchovávat unikátní název výrobku, počet kusů, jež je skladem a jeho EAN, což je mezinárodní číslo obchodní položky, které bývá nejčastěji tištěno i ve formě čárových kódů. Tento kód může mít 8,12,13 nebo 14 znaků.

```
CREATE TABLE Vyrobek(
idVyrobku int IDENTITY,
NazevVyrobku VARCHAR(50) NOT NULL UNIQUE,
EAN VARCHAR(14) NOT NULL UNIQUE,
pocetKS int NOT NULL CHECK(pocetKS >= 0)
PRIMARY KEY (idVyrobku),
CONSTRAINT check_EAN CHECK(EAN like
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR EAN like
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR EAN like
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9]'
OR EAN like
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9]') );
```

Primárním klíčem této tabulky by mohl být právě kód ean, který je pro každý výrobek unikátní, nicméně v celém konceptu jsou využívány umělé primární klíče. Primární klíč je zde definován zvlášť, což je alternativou ke klasickému použití klíčového slova přímo při definici atributu.

Složený primární klíč za využití integritního omezení by v tomto případě mohl vypadat například takto:

```
CONSTRAINT pk_Vyrobek PRIMARY KEY (EAN, NazevVyrobku);
```

Počet kusů jednotlivých výrobků nesmí klesnout pod hodnotu 0 což je zajištěno opět integritním omezením a kód ean je za pomoci regulárních výrazů přímo definován a jakákoliv jiná hodnota bude označena za nepřípustnou stejně jako hodnota opakující se. V atributu EAN nejsou též přípustné jiné než číselné hodnoty navzdory tomu, že je definován jako varchar.

**Tabulka Odběratel**

Dle požadavků firmy může být odběratelem i fyzická osoba. Proto je položka IČO nepovinná a jméno a příjmení osoby se uvede v položce JmenoOdberatele. Problematiku rozlišení fyzické a právnické osoby popřípadě firmy by bylo vhodné rozlišit v aplikaci nabídnutím odlišných kolonek pro vyplnění na základě volby mezi právnickou a fyzickou osobou. Data by poté aplikace vyhodnotila a do databáze poslala vhodným způsobem. Nepovinná je i položka PSC, neboť běžně k identifikaci adresy stačí ulice, číslo popisné a město. Identifikační číslo osoby vyjadřuje jedinečné číslo právnické osoby, podnikající fyzické osoby nebo organizační složky státu. V naší databázi je uloženo jako číselná hodnota integer a neuvádí se tedy nuly potřebné pro doplnění na osmimístné číslo platné dle českých zákonů. Pro potřeby fakturace by bylo třeba doplnit eventuální chybějící nuly, které je zbytečné uchovávat v databázi.

```
CREATE TABLE Odberatel(
idOdberatele int IDENTITY PRIMARY KEY,
JmenoOdberatele VARCHAR(50) NOT NULL,
ICO int UNIQUE, Mesto VARCHAR(30) NOT NULL,
Ulice VARCHAR(50) NOT NULL,
CisPop smallint NOT NULL CHECK(Cispop > 0),
PSC int,
TelKontakt VARCHAR(14) NOT NULL UNIQUE,
CONSTRAINT check_ICO CHECK(ICO BETWEEN 0 AND 99999999),
CONSTRAINT check_PSC CHECK(PSC BETWEEN 1 AND 99999),
CONSTRAINT check_TelKontakt CHECK( TelKontakt like
'[+][1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9]0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9]') );
```

Použitá integritní omezení znemožňují zadat záporné číslo popisné a určují správný rozsah pro ičo a psč. Telefonní číslo je za pomoci regulárních výrazů ošetřeno pro zadání mezinárodních předvoleb a následného čísla. Telefonní číslo a ičo jsou unikátní údaje, což vyjadřuje omezení klíčovým slovem UNIQUE.

**Tabulka Zaměstnanec**

Evidence zaměstnance obsahuje základní údaje o zaměstnancích, které podnik dále využívá pro interní účely. Tabulka obsahuje v té době ještě nepřiřazenou hodnotu, které bude využita pro dodatečné zařazení zaměstnance dle jeho náplně práce k příslušnému oddělení. Jelikož tabulka oddělení ještě není vytvořena, není možné se odkazovat na neexistující tabulku. Při pokusu o takovouto činnost SQL server vrací chybovou hlášku o nevalidní referenci.

```
CREATE TABLE Zamestnanec(
idZamestnance int IDENTITY PRIMARY KEY,
Jmeno VARCHAR(45) NOT NULL,
Prijmeni VARCHAR(45) NOT NULL,
RodCis VARCHAR(11) NOT NULL UNIQUE,
Mesto VARCHAR(30) NOT NULL,
Ulice VARCHAR(50) NOT NULL,
CisPop smallint NOT NULL CHECK(CisPop> 0),
PSC int, Plat int NOT NULL CHECK(Plat> 0),
TelKontakt VARCHAR(14) NOT NULL UNIQUE,
OddeleniID int,
CONSTRAINT check_PSC_zam CHECK(PSC BETWEEN 1 AND 99999),
CONSTRAINT check_rodcis CHECK(RodCis like
'[0-9][0-9][0-9][0-9][0-9][0-9][/][0-9][0-9][0-9][0-9]'),
CONSTRAINT check_TelKontakt_zam CHECK( TelKontakt like
'[+][1-9][2-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][2-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][0-9][2-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][0-9][0-9][2-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9]') );
```

Rodné číslo je vymezeno doménovou integritou za použití regulárního výrazu. Psč je vymezeno rozsahově a číslo popisné spolu s platem zaměstnance nepovoluje zápornou hodnotu díky CHECK omezení. Telefonní kontakt je omezen totožně jako u odběratele a stejně tak i u následné tabulky oddělení.

**Tabulka Oddělení**

Oddělení má svůj specifický, jedinečný název a telefonní číslo a je přímo spjato s tabulkou zaměstnanců, přičemž platí pravidlo evidence vedoucího zaměstnance pro dané oddělení. Toto pravidlo je vynuceno referenční integritou za použití klíčových slov FOREIGN KEY.

```
CREATE TABLE Oddeleni(
idOddeleni int IDENTITY PRIMARY KEY,
NazevOdd VARCHAR(50) UNIQUE NOT NULL,
TelKontakt VARCHAR(14) UNIQUE NOT NULL,
VedouciZamID int UNIQUE NOT NULL,
FOREIGN KEY (VedouciZamID)
REFERENCES Zamestnanec(idZamestnance),
CONSTRAINT check_TelKontakt_oddeleni CHECK( TelKontakt like
'[+][1-9][2-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][2-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][0-9][2-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9]'
OR TelKontakt like
'[+][1-9][0-9][0-9][0-9][2-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9]') );
```

Vazbu na zaměstnanecké oddělení vymezuje navíc klauzule UNIQUE, která zamezuje dvojímu vedení jednoho oddělení, stejně tak není možné zadat oddělení se stejným telefonním číslem a názvem.

Po vytvoření této tabulky se můžeme navrátit k nevyužitému atributu v tabulce zaměstnanec a dodatečně vytvořit referenci a s ní spojenou referenční integritu.

```
ALTER TABLE Zamestnanec ADD FOREIGN KEY (OddeleniID)
REFERENCES Oddeleni(idOddeleni) ON DELETE SET NULL;
```

Při odstranění oddělení je indikátor nastaven na nulu. Po vložení všech přístupných oddělení by bylo též vhodné nastavit nutnost přiřazení zaměstnance do nějakého oddělení. Složitější integritní omezení je třeba zapracovat dodatečně a s pomocí SQL funkce, neboť integritní omezení při své tvorbě neumožňuje tvorbu a využití pod dotazů přímo.

```
CREATE FUNCTION myCheckPlat(@IDzam int)
RETURNS int
AS
BEGIN
IF EXISTS(
SELECT idZamestnance FROM Zamestnanec
WHERE @IDzam NOT IN (SELECT VedouciZamID FROM Oddeleni))
return (
SELECT MIN(Plat) FROM Zamestnanec
WHERE idZamestnance IN (SELECT VedouciZamID FROM Oddeleni))
return 500 000;
END
GO
```

Tato SQL funkce vyhodnotí na základě vstupního identifikátoru zaměstnance zda-li zaměstnanec je, nebo není vedoucí. V případě, že je funkce vrátí platové omezení 500000, což by vyznačovalo úpravu dat, jelikož není možné vytvořit přímo vedoucího pracovníka. Pokud není, návratovou hodnotou je nejmenší platové ohodnocení mezi vedoucími pracovníky.

```
ALTER TABLE Zamestnanec
ADD CONSTRAINT check_plat_vedouci
CHECK(Plat < dbo.myCheckPlat(idZamestnance));
```

Aplikací tohoto integritního omezení pak zabráníme nelogickému ohodnocení platových podmínek zaměstnance, který není ve vedení, nemůže mít tak větší plat než někdo z vedoucích pracovníků.

#### Tabulka Služební cesta

Tabulka služební cesta slouží k plánování služebních cest. Cesta má své datum konání, plánovaný čas odjezdu a příjezdu a stručný popis obsahující důvod cesty.

```
CREATE TABLE SluzebniCesta(
idSluzCesty int IDENTITY PRIMARY KEY,
DatumCesty DATE NOT NULL,
CasOdjezdu TIME(0) NOT NULL,
CasPrijezdu TIME(0)NOT NULL,
PopisCesty VARCHAR(100) NOT NULL,
CONSTRAINT Odjezd_prijezd
CHECK(CasPrijezdu >= DATEADD(hour,2,CasOdjezdu)) );
```

Doménová integrita je zde řešena za pomocí klauzule CHECK a funkce na úpravu času respektive data. Toto omezení zajišťuje, že služební cesta je vypsána alespoň na dvě hodiny a čas návratu ze služební cesty je tedy alespoň dvě hodiny po odjezdu.

**Tabulka Zaměstnanec na cestě**

Tato tabulka je pomocnou tabulkou pro evidenci zaměstnanců, kteří jedou na nějakou služební cestu, a také zajišťuje evidenci zaměstnanců účastnící se služební cesty. X zaměstnanců se může zúčastnit Y služebních cest. Propojení je zajištěno referenční integritou.

```
CREATE TABLE ZamNaCeste(
idZamNaCeste int IDENTITY PRIMARY KEY,
SluzebCestaID int NOT NULL,
ZamestnanecID int NOT NULL,
FOREIGN KEY (ZamestnanecID)
REFERENCES Zamestnanec(idZamestnance),
FOREIGN KEY (SluzebCestaID)
REFERENCES SluzebniCesta(idSluzCesty),
CONSTRAINT check_Unique_ZamNaCeste
UNIQUE(SluzebCestaID, ZamestnanecID) );
```

Referenční integrita je zde podpořena vymezením unikátní kombinace identifikátorů služební cesty a zaměstnance, tedy žádný zaměstnance nemůže jet na stejnou služební cestu dvakrát. Dodatečné omezení vyžaduje implementaci další SQL funkce, která vrátí počet narušení stanoveného pravidla. Vrácená hodnota 0 odpovídá neporušení daného pravidla a umožní, při použití této funkce v integritním omezení, vložení nového záznamu. V opačném případě je vložení zakázáno s hláškou narušení integrity.

```
CREATE FUNCTION myCheckZamestnanecNaCeste()
RETURNS int
AS
BEGIN
RETURN(SELECT COUNT(*) FROM
(SELECT * FROM SluzebniCesta
JOIN ZamNaCeste
ON SluzebniCesta.idSluzCesty = ZamNaCeste.SluzebCestaID)
AS a
JOIN (SELECT * FROM SluzebniCesta2
JOIN ZamNaCeste
ON SluzebniCesta.idSluzCesty = ZamNaCeste.SluzebCestaID)
AS b
ON a.SluzebCestaID <> b.SluzebCestaID
AND a.CasOdjezdu < b.CasPrijezdu
AND a.CasPrijezdu > b.CasOdjezdu
AND a.ZamestnanecID = b.ZamestnanecID
AND (SELECT SluzebniCesta.DatumCesty
FROM SluzebniCesta
WHERE SluzebniCesta.idSluzCesty = a.SluzebCestaID) =
(SELECT SluzebniCesta.DatumCesty
FROM SluzebniCesta
WHERE SluzebniCesta.idSluzCesty = b.SluzebCestaID))
END
GO
```

Funkce porovná propojená data mezi sebou, nikoliv však data stejná, pouze data rozdílná. Využitím propojení na základě zaměstnance, který jede na konkrétní služební cestu s danou služební cestou, můžeme porovnat časy příjezdu a odjezdu v dni konání služební cesty.

```
ALTER TABLE ZamNaCeste
ADD CONSTRAINT check_zamestnanec_na_ceste
CHECK(dbo.myCheckZamestnanecNaCeste() = 0);
```

Tímto využitím zabráním zaměstnanci, aby se zůčastnil služební cesty, která se kryje s jinou jeho služební cestou v daný den.

**Tabulka Objednávka**

Evidence objednávek je pro podnik klíčovým faktorem. Primárním klíčem by mohlo být číslo faktury, ale opět se držíme umělých klíčů. Reference je zde na zaměstnance, který objednávku přijal a na odběratele, který si výrobky firmy objednal. Objednávkový formulář by byl následně realizován pohledem, nebo skrze aplikaci, která by běžela nad databází. Pohled by byl kombinací právě objednávky a tabulky pomocné, která eviduje objednané výrobky a je popsána níže. Formulář v interní aplikaci podniku by pak vložil údaje do obou tabulek, což je schůdnější varianta a vyhneme se tak přílišné duplikaci dat.

```
CREATE TABLE Objednavka(
idObjednavky INT IDENTITY PRIMARY KEY,
cisloFaktury INT UNIQUE NOT NULL CHECK(cisloFaktury> 0),
DatumSplatnosti DATE NOT NULL,
ZpracovanoZamestnancemID INT NOT NULL,
OdberatelID INT NOT NULL,
FOREIGN KEY (ZpracovanoZamestnancemID)
REFERENCES Zamestnanec(idZamestnance),
FOREIGN KEY (OdberatelID)
REFERENCES Odberatel(idOdberatele) );
```

Integritním omezením zamezujeme vložení záporné hodnoty do atributu cisloFaktury a tento atribut je též definován jako unikátní. Omezení referenční integrity pro zúžení zaměstnanců, kteří mohou zpracovat objednávku, využívá následující funkci.

```
CREATE FUNCTION myCheckZam(@ZpracovanoZamestnancemID int)
RETURNS VARCHAR(5)
AS
BEGIN
IF EXISTS (
SELECT OddeleniID FROM Zamestnanec
WHERE idZamestnance = @ZpracovanoZamestnancemID
AND OddeleniID = (
SELECT idOddeleni FROM Oddeleni
WHERE NazevOdd = 'obchodni'))
return 'True'
return 'False'
END
```

```
GO
```

Funkce na základě vstupního parametru vrátí hodnotu TRUE nebo FALSE, je-li daný zaměstnanec členem obchodního oddělení nebo nikoliv. Použitím funkce pro doménovou integritu takto zabráním vložení objednávky, kterou by se pokoušel zpracovat nevhodný zaměstnanec.

```
ALTER TABLE Objednavka
ADD CONSTRAINT zamestnanec_obchodniho_odd
CHECK(dbo.myCheckZam(ZpracovanoZamestnancemID)='True');
```

**Tabulka pomocná k objednávce**

Jedná se o pomocnou tabulku, která se využívá k evidenci objednaných výrobků a k evidenci počtu kusů, které si odběratel objednal. Objednávka smí obsahovat více výrobků a výrobek smí být objednán nespočetněkrát, tedy dokud je na skladě. Problém tohoto rázu řeší právě pomocná tabulka.

```
CREATE TABLE pomTabObjednavka(
idObjednavkovyList int IDENTITY PRIMARY KEY,
VyrobekID int NOT NULL,
pocetKS smallint CHECK(pocetKS > 0) DEFAULT (1),
ObjednavkaID int NOT NULL,
FOREIGN KEY (VyrobekID)
REFERENCES Vyrobek(idVyrobku),
FOREIGN KEY (ObjednavkaID)
REFERENCES Objednavka(idObjednavky) );
```

Pokud je vyplněn počet kusů objednaného výrobku, tak toto číslo nesmí být menší než 0. V případě nevyplnění je nastavena defaultní hodnota jednoho kusu. Vhodným omezením je též nemožnost objednat výrobek, který již není skladem, což řeší dodatečně přidané integritní omezení využívající následnou funkci.

```
CREATE FUNCTION myCheckPocetKS(@VyrobekID int)
RETURNS int
AS
BEGIN
return (SELECT pocetKS FROM Vyrobek
```

```
WHERE idVyrobku = @VyrobekID)
END
GO
```

Funkce vrací na základě vstupního parametru počet kusů daného výrobku.

```
ALTER TABLE pomTabObjednavka
ADD CONSTRAINT limit_objednanych_vyrobku
CHECK(pocetKS <= dbo.myCheckPocetKS(VyrobekID));
```

Aplikací tohoto pravidla zamezujeme objednání více kusů výrobků, než máme na skladě.

### 7.2.3 Triggery

Provedení změn v jiné tabulce na základě změny v jedné je vhodné vzhledem k ulehčení práce. Ušetří se tak zejména čas strávený nad úpravou dat, nebo přidávání dat ručně a předejde se tak možným chybám při úpravě potřebných položek. Triggery v obou případ mají definovány své interní proměnné pro větší přehlednost a snazší orientaci. Samotný SQL dotaz pak nepůsobí jako zhůvěřilost a působí jednoduše a hlavně zastane svou funkci. Následující trigger zvyšuje platové ohodnocení zaměstnance, který byl vybrán pro vedení nově vzniklého oddělení.

```
CREATE TRIGGER Trigger_Update_Plat_onInsert
ON  Oddeleni
AFTER INSERT
AS
DECLARE @idZam int;
DECLARE @idOdd int;
SELECT @idZam = i.VedouciZamID FROM inserted i;
SELECT @idOdd = i.idOddeleni FROM inserted i;

BEGIN
UPDATE Zamestnanec
SET Plat = Plat * 1.8,
OddeleniID = @idOdd
WHERE idZamestnance = @idZam;
```

```
PRINT 'AFTER INSERT
Trigger_Update_Plat_onInsert dokončen.'
END
GO
```

Tento trigger přiřadí vybranému vedoucímu oddělení a vynásobí jeho plat hodnotou 1,8 a vypíše hlášku o úspěšném dokončení. Obdobně by se dalo přistoupit i k změně ve vedení. Takový trigger by reagoval ne na INSERT, ale na UPDATE a obdobně by přiřadil nově zvolenému zaměstnanci platové ohodnocení a původnímu vedoucímu by platové ohodnocení snížil.

Následující trigger řeší automatické odečtení objednaných výrobků. Stav výrobků po objednávce tedy bude menší právě o počet objednaných výrobků. Nedostatkem kusů na skladě se nemusí zaobírat, neboť to již vyřešilo integritní omezení použité během návrhu této tabulky.

```
CREATE TRIGGER Trigger_Update_PocetKsVyrobku_onInsert
ON pomTabObjednavka
AFTER INSERT
AS
DECLARE @idVyrobku int;
DECLARE @pocetKSobjednanych int;
SELECT @idVyrobku = i.VyrobekID FROM inserted i;
SELECT @pocetKSobjednanych = i.pocetKS FROM inserted i;

BEGIN
UPDATE Vyrobek
SET pocetKS = (pocetKS - @pocetKSobjednanych)
WHERE idVyrobku = @idVyrobku;
PRINT 'AFTER INSERT
Trigger_Update_PocetKsVyrobku_onInsert dokončen.'
END
GO
```

## 7.3 Shrnutí

Databázi jako takovou tvoří množina komplexním problémů, které jsou součástí jejího návrhu a provozu, což nebylo součástí této práce. Nastínené řešení představuje ukázku možného řešení problematiky integritních omezení, které je podpořeno SQL funkcemi. Práce ukazuje konkrétní použití SQL a aplikace integritních omezení přímo při tvorbě tabulek využitím klíčových slov a klauzulí, které mají svou obecnou funkci, a jejich kombinací lze docílit rozličných omezení. Obecně platná je použitá entitní i referenční integrita spolu s ukázkou možných dalších zúžení povolených vstupních dat.

Doménová integrita v tomto řešení demonstruje využití SQL funkcí a obecně platný princip nemožnosti použití sub dotazů pro klauzuli CHECK. Využitím triggerů usnadňujeme práci uživateli, eventuálně aplikaci, která nad databází běží. Integritní omezení nabízí vhodné řešení problémů konzistence dat, jelikož fungují jak na nadstavbě nad databází tak i interně, pokud by SQL administrátor doplňoval data přímo.

# 8 Závěr

Celá práce rozebírá problematiku datové integrity v relačně databázovém zpracování dat a stav v jakém se v současnosti nachází. Vymezuje teoreticky i prakticky použití těchto omezení a demonstruje jejich sílu a praktické využití pro řešení problémů. Výsledkem práce bylo uvedení integritních omezení v provoz a získání konzistentních a přesných dat, která tak zjednoduší práci s nimi a napomůže tak k realizaci přesných vztahů mezi nimi a jejich souvislostmi, což jsem demonstroval na podniku s typickou databázovou charakteristikou.

Použitá metodika zadané bakalářské práce byla založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Stěžejní byly metody a techniky relačně databázové technologie ve spojení s problematikou datové integrity. Navrhované řešení zohledňuje požadavky klienta a je spojené s řešenou záležitostí. Na základě syntézy teoretických poznatků a dosažených výsledků jsou formulovány závěry této bakalářské práce.

Přínosem tohoto řešení je okamžitá aplikace omezení již při tvorbě databáze respektive při tvorbě celého řešení. Systém s kvalitní datovou základnou disponuje stabilitou a snadněji se udržuje a z takovýchto dat může profitovat každá aplikace běžící nad nimi. Přínosem integritních omezení je jejich vynucení již při tvorbě databáze, takže je zajištěna správnost dat již při prvotním plnění a propojení s klientem. Triggery poté umožňuji částečnou automatizaci plnění databáze daty, jelikož jsou navázány na existující procesy podniku. Může se jednat o automatické zanesení objednávky do databáze podle EAN. Výhodou je i možnost kontroly přístupu k datům samotným. V případě potřeby je možné přidat nová integritní omezení, která se aplikují i zpětně na již existující data. Je tedy možné data ještě dodatečně zpřesnit a objevit zpětně narušení nově přidaných pravidel.

Aplikace integritních omezení ulehčuje práci nástavbové aplikaci, která se tak může věnovat důležitějším činnostem z hlediska dané aplikace. Správná a validní data neulehčují práci jen programátorům, ale zejména uživatelům, kteří mohou výstupní data dále využívat a těžit z jejich korektnosti. Relevantní a korektní data jsou klíčem ke správnému fungování systému a ve své podstatě by data nerelevantní a nekorektní bylo zbytečné ukládat, jelikož data s nulovou vypovídající hodnotou jsou bezcenná. Vynucování těchto pravidel vyžaduje komplexní myšlení, znalost SQL, databází a programování. Přináší však své výsledky a měla by být nedílnou součástí při tvorbě databáze. Navíc při dnešním širokém využívání internetu, různých portálů a aplikací ať již mobilních či standartních, jsou data základem úspěchu a jsou též klíčem pro získávání informací. Správně formátovaná, přesná a korektní data zaručují při velkém objemu snadnější dohledatelnost a přehlednost těchto dat.

Použitá integritní omezení v praktické části jsou ukázkou využití integrity a zabezpečení

přesných a konzistentních dat. Tyto principy jsou obecně platné a po úpravě pro daný problém jsou použitelné v širším měřítku. Toto řešení lze tedy využít jako jakýsi návod pro práci s integritními omezeními.

Závěrem je třeba zmínit, že integrita dat je nutností, ale vznáší vysoké kvalifikační požadavky. Proto je třeba k ní přistupovat odpovědně, svědomitě a pozorně.

# Literatura


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

[2] HERNANDEZ J. Michael. Návrh databází. Praha: Grada, 2006. ISBN 80-247-0900-7.

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

[4] ELMASRI Ramez a NAVATHE B. Shamkant. Fundamentals of Database Systems (6th Edition). Adison Wesley, 2010. ISBN 978-0-136-08620-8.

[5] HAAN L., FINK D., GORMAN T., JØRGENSEN I., MORTON K.. Beginning Oracle SQL. Berkeley: Apress, 2009. ISBN 978-1-4302-7197-0.

[6] SHARMA N., PERNIU L., CHONG R. F., IYER A., NANDAN Ch., MITEA A., NONVINKERE M., DANUBIANU M.. Database fundamentals. IBM, 2010. Dostupné z: http://public.dhe.ibm.com/software/dw/db2/express-c/wiki/Database_fundamentals.pdf

[7] GREFEN Paul a Peter APERS Integrity control in relational database. Enschede: University of Twente, 1993. Dostupné z: http://core.ac.uk/download/pdf/11454268.pdf

[8] DATE C.J.. An introduction to database systems. Pearson Academic Computing, 2004. ISBN 0-321-18956-6.

[9] SANDHU R. a S. JAJODIA. Integrity mechanisms in database management systems. 1990. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.2380

[10] JANSSEN Dale a Cory JANSSEN. TECHOPEDIA [online]. JANSSEN Cory, ©2010-2015 [cit. 2014-9-13]. Dostupné z: http://www.techopedia.com/definition/7272/foreign-key

[11] MANUALY [online]. TeorieDB. ©2005-2006 [cit. 2015-3-4]. Dostupné z: http://www.manualy.net/article.php?articleID=15

[12] ALECHINA Natasha. The relational model [online prezentace]. Nottingham: University of Nottingham, 2009 [cit. 2014-11-4]. Dostupné z: http://www.cs.nott.ac.uk/~nza/G51DBS08/lecture3.ppt

[13] ALECHINA Natasha. Entity/Relationship modelling [online prezentace]. Nottingham: University of Nottingham, 2009 [cit. 2015-3-4]. Dostupné z: http://www.cs.nott.ac.uk/~nza/G51DBS08/lecture4.ppt

[14] ETUTORIALS [online]. Validation tables. ©2008-2015 [cit. 2014-10-23]. Dostupné z: http://etutorials.org/SQL/Database+design+for+mere+mortals/Part+II+The+Design+Process/Chapter+11.+Business+Rules/Validation+Tables/

[15] ORACLE [online]. Oracle Database Online Documentation, 10g Release 2. ©2015 [cit. 2015-1-17]. Dostupné z: http://docs.oracle.com/cd/B19306_01/server.102/b14220/data_int.htm

[16] ORACLE [online]. Oracle7 Server Concepts Manual, Database triggers. ©1996 [cit. 2015-1-17]. Dostupné z: http://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm

[17] ANSI [online]. ANSI: Historical Overview. ©2015 [cit. 2015-1-23]. Dostupné z: http://www.ansi.org/about_ansi/introduction/history.aspx?menuid=1

[18] ORACLE [online]. Oracle Big Data SQL. 2014-7-15 [cit. 2015-1-23]. Dostupné z: http://www.oracle.com/us/corporate/pressrelease/big-data-SQL-071514

[19] Database availability and integrity in NoSQL [online prezentace]. Slideshare: FIRDAUSILLAH Fahri 2012-1-31 [cit. 2015-1-23]. Dostupné z: http://www.slideshare.net/kaqfa/noSQL-availability-integrity

[20] Hans Vestberg. ERICSSON: Truth not trust – the importance of data integrity in the Networked Society [online]. Geoff Hollingworth 2014-6-20 [cit. 2015-3-4]. Dostupné z: http://www.ericsson.com/thinkingahead/the-networked-society-blog/2014/06/20/truth-trust-importance-data-integrity-networked-society/

# Seznam obrázků