Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Zabezpečení relačně databázových evidencí

Jindřich Novotný

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jindřich Novotný

Informatika

Název práce

Zabezpečení relačně databázových evidencí

Název anglicky

Security of relational database records

Cíle práce

Bakalářská práce je zaměřena na problematiku zabezpečení databázově evidovaných dat v podnikatelských subjektech. Hlavním cílem této práce bude:
a) objasnit teoretické principy relačně databázové technologie v kontextu s problematikou zabezpečení db evidovaných dat,
b) zmapovat momentální stav této problematiky a vymezit její relevantnost včetně požadavků na ni kladených,
c) navrhnout řešení této problematiky v souladu s identifikovanými požadavky a se zřetelem na reálné možnosti podnikatelských subjektů,
d) ověřit funkčnost navrhovaného řešení v rámci funkčního prototypu,
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. Stěžejní pro vypracování této závěrečné práce budou především metody a techniky relačně databázové technologie v kontextu s problematikou zabezpečení databázově koncipované datové základny. Navrhované řešení bude zohledňovat identifikované požadavky a očekávání spojená s řešenou problematikou. Na podkladě syntézy teoretických poznatků a dosažených výsledků budou formulovány závěry této bakalářské práce a následně zobecněny pro další možná použití.

Doporučený rozsah práce

40-55 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 978-80-251-2617-2LONEY, K.: Oracle Database, kompletní průvodce. Computer press Brno 2010. ISBN 978-80-251-2489-5

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 9. 3. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 09. 03. 2015

Čestné prohlášení

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

V Praze dne 16.3.2015 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Václavu Vostrovskému, Ph.D. za vedení mé bakalářské práce, za veškeré odborné rady, jež mi poskytl a za čas, který mi věnoval během přípravy bakalářské práce.

Zabezpečení relačně databázových evidencí

Security of relational database records

Souhrn

Cílem této bakalářské práce je objasnit teoretické principy relačně databázové technologie v kontextu s problematikou zabezpečení podnikových databází. Zmapovat aktuální stav a vymezit relevantnost dané problematiky. Na základě teoretických poznatků pak navrhnout konkrétní řešení. K řešení zvoleného problému je v této práci využita kombinace rolí a pohledů, jež slouží k tvorbě interního zabezpečení, dále pak profily, které jsou naopak používány při tvorbě externí bezpečnosti. Řešení vytvořené v rámci této práce napomáhá snížení rizika přístupu neautorizovaných osob k datům uloženým v relačních databázích, a to především díky přesnému vymezení kompetencí jednotlivých uživatelů a posílení bezpečnosti jejich účtů. Přínosem této práce je pak prokázání nutnosti vytváření bezpečných databázových systémů a porovnání jednotlivých technik, jež jsou k tomuto účelu nejčastěji využívány. Navržené řešení lze navíc použít jako návod, jak k problematice zabezpečení relačně databázových evidencí přistupovat.

Klíčová slova

relačně databázová technologie, oprávnění, role, pohledy, profily, SQL

Summary

The goal of this thesis is to clarify the theoretical principles of relational database technology in context with the issue of the security of enterprise databases. To map the current state and determine the relevance of the issue. Then design a specific solution based on theoretical knowledge. To solve the issue is used a combination of roles and views, that is used to create an internal security as well as profiles, which on the other hand are used in the creation of an external security. Solution developed as a part of this thesis helps to reduce the risk of unauthorized persons access to data stored in relational databases, mainly thanks to a precise definition of the competences of individual users and strengthen the security of their accounts. The contribution of this thesis is to demonstrate the need to create a secure database systems and comparison of techniques that are most commonly used for this purpose. The proposed solution can also be used as a guide describing how to access the issue of security of relational database records.

Keywords:

relational database technology, permissions, roles, views, profiles, SQL

Obsah

1 Úvod

Tématem bakalářské práce je zabezpečení relačně databázových evidenci. Daná problematika je v dnešní době velice důležitá zejména proto, že velké množství existujících systémů pracuje právě na základě relačně databázové technologie. Tyto systémy jsou pak velmi často v režii podnikatelských subjektů, jež ve svých databázích uchovávají důležitá data, která je třeba účinně chránit před neautorizovanými osobami. Ať už se jedná o citlivá data osob, jako například klientů či zaměstnanců, jež z právního hlediska podléhají utajení a jejichž zveřejnění by mohlo vést k právním krokům proti organizaci. Nebo o data strategického významu, jako jsou nejrůznější výrobní postupy, technologické specifikace či informace získané z průzkumů trhu. Informace jsou pro podnikatelské subjekty zdrojem konkurenceschopnosti a data, jež jsou nositelem informací, by měla být velmi dobře chráněna, chce-li společnost prosperovat.

S řešením problematiky zabezpečení relačně databázových evidencí je možné se velmi často setkat u podniků zveřejňujících data na internetu. V tomto prostředí je totiž možné databáze nejsnáze napadnout, a tak je třeba, aby databáze takovýchto podnikatelských subjektů byly kvalitně zabezpečeny. Zdaleka nejčastěji se však řešení zabezpečení relačních databází uplatní u podnikatelských subjektů ve finančním a především pak v bankovním sektoru. Instituce v těchto sektorech většinou pracují s obrovským množstvím citlivých údajů o svých klientech, a proto je na místě, že zabezpečení svých databází vynakládají odpovídající finanční a časové prostředky.

V rámci třetí kapitoly podrobněji rozebereme proč je potřeba řešit zabezpečení relačně databázových evidencí, objasníme, co danou problematiku definuje a jaké nároky jsou na ni kladené. Následně v kapitole čtvrté objasníme základní principy relačně databázové technologie, které napomáhají vytvářet stabilní a bezpečné databáze. V kapitole páté se pak hlouběji ponoříme do problematiky zabezpečení, definujeme, před čím konkrétně je databáze třeba chránit a představíme si nástroje relačně databázové technologie, jež jsou k řešení této problematiky využívány. V šesté kapitole poté navážeme na teoretické základy získané v předchozích kapitolách a pro fiktivní společnost vytvoříme databázi, na níž budou formou ukázek uplatněny získané poznatky. Na závěr pak shrneme jednotlivé poznatky, které vyvstanou v průběhu zpracování této bakalářské práce.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je zaměřena na problematiku zabezpečení relačně databázových evidencí v podnikatelských subjektech. Hlavními cíli této práce pak je zmapovat momentální stav řešené problematiky a vymezit její relevantnost včetně požadavků na ni kladených. Objasnit teoretické principy relačně databázové technologie v kontextu s problematikou zabezpečení databázově evidovaných dat. Na základě teoretických poznatků pak navrhnout a ověřit řešení dané problematiky v souladu s identifikovanými požadavky a se zřetelem na reálné možnosti podnikatelských subjektů. Získané poznatky následně zobecnit pro další možná využití.

2.2 Metodika

Metodika zpracování této bakalářské práce je založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Hlavními informačními zdroji pak budou především odborné publikace pojednávající o problematice relačně databázové technologie a dokumentace Oracle. Stěžejní pro vypracování této závěrečné práce budou metody a techniky relačně databázové technologie využívané v kontextu s problematikou zabezpečení relačních databází podnikatelských subjektů. 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.

3 Relevantnost problematiky zabezpečení

Potřeba zabezpečení informací, respektive dat, sahá hluboko do historie lidstva. Již v dobách, kdy lidé začali informace shromažďovat a zapisovat na fyzická média, začaly vznikat problémy a vyvstávat otázky, s nimiž je nutné se vypořádat i dnes ve dvacátém prvním století. Velké množství dnes existujících systémů, v nichž je třeba uchovávat a zpracovávat velké objemy dat, pracuje na základě relačně databázové technologie. Proto je třeba zkoumat právě problematiku zabezpečení relačně databázových evidencí. Tato problematika je dnes mnohem aktuálnější než v minulosti, což je způsobeno především následujícími faktory.

1. Společnost dnes generuje mnohem větší objem dat, než tomu bylo kdy dříve, a velké množství z těchto dat je možné kvalifikovat jako data citlivá či data strategicky významná pro podnikatelské subjekty a jako taková by měla podléhat utajení.[8]
2. Velké množství dat je zprostředkováváno prostřednictvím internetu, kde se k nim ovšem, pokud by nebyla efektivně zabezpečena, může dostat prakticky kdokoli, což je značně nežádoucí, jedná-li se o data podléhající utajení.[5]
3. V dnešní tržně ekonomické společnosti jsou informace jedním ze základních zdrojů konkurenceschopnosti každého podnikatelského subjektu, a tudíž aby mohla jakákoli firma prosperovat, musí umět informace nejen rychle získávat a efektivně využívat, ale také musí být schopna tyto informace ochránit, aby neztratila konkurenční výhodu.

Ze zmíněných faktů vyplývá, že řešení problematiky zabezpečení nalezne uplatnění nejčastěji u databází, které jsou veřejně dostupné například prostřednictvím internetu a v nichž jsou evidována citlivá či strategicky významná data. Evidence odpovídající zmíněným parametrům jsou převážně vlastnictvím podnikatelských subjektů. Může se jednat o nejrůznější typy podniků, avšak nejčastěji jde o společnosti podnikající ve finančním a především pak v bankovním sektoru. Instituce v těchto sektorech vždy pracují s velkým množstvím citlivých údajů o svých klientech, a tak musí na zabezpečení svých databází vynakládat nemalé finanční a časové prostředky.[5]

V případech, kdy dojde k zanedbání při tvorbě bezpečnostních opatření nebo k úplné absenci jakéhokoli řešení bezpečnosti, mohlo by dojít k poškození, ztrátě či dokonce ke zcizení dat, což by zejména u podnikatelských subjektů mohlo vést ke ztrátě důvěryhodnosti, konkurenceschopnosti nebo v krajních případech i k právním krokům proti dané společnosti.[2] Zmíněné následky, jimž by mohla společnost čelit v případě zanedbání bezpečnosti, včetně konkrétních bezpečnostních rizik, s nimiž se při řešení zabezpečení lze běžně setkat, jsou podrobněji popsány v části 5.1.

Doposud bylo objasněno, k jakým následkům může dojít, jestliže zanedbáme řešení bezpečnosti a také, kde si tato problematika nalezne své uplatnění. Zbývá tedy definovat, jaké požadavky jsou kladeny na řešení dané problematiky a s jakými záležitostmi se vlastně v souvislosti s řešenou problematikou musíme vypořádat. Lze je například popsat několika otázkami, kterými je třeba se při tvorbě bezpečnosti zabývat a jež danou problematiku vystihují. Jak zabránit v přístupu k důležitým informacím neautorizovaným osobám? Jakým způsobem rozlišit, kdo je autorizován k jakým činnostem při práci s konkrétními daty? Jak zajistit důsledné dodržování zásad bezpečnosti, jež snižují riziko napadení databáze za použití cizího účtu? Jak co nejvíce zmírnit riziko vzniku chyb v důsledku lidské činnosti, jež mohou vést k poškození dat? Jak vytvořit kompromis mezi bezpečností a uživatelským komfortem, abychom získali bezpečné prostředí, které však zároveň příliš neomezuje své uživatele a nesnižuje tak jejich pracovní efektivitu? Na tyto ale i další otázky je nutné nalézt odpovědi, chceme-li vytvořit co možná nejbezpečnější systém.

Díky využití relačně databázové technologie je možné na zmíněné otázky nalézt relevantní odpovědi. Ty existují ve formě využití nástrojů, jež nám relační databáze poskytují. Především jde o využití oprávnění a rolí, pohledů a v neposlední řadě též uživatelských účtů a profilů. Zmíněné nástroje jsou primárními prostředky, jež jsou využívány v souvislosti s problematikou zabezpečení relačně databázových evidencí.[6]

Aby však vůbec mělo význam začít řešit zabezpečení, je nejprve třeba zajistit, že navrhovaná databáze bude robustní a spolehlivá, díky čemuž se stane stabilní platformou, na níž lze začít tvořit jednotlivé bezpečnostní prvky. K zajištění spolehlivosti databáze slouží prvky relačně databázové technologie, jako jsou transakční zpracování, integritní omezení, primární klíče a samotné vlastnosti relačních tabulek.

4 Teoretické principy relačně databázové technologie

Relačně databázová technologie je jednou z hlavních součástí většiny moderních informačních systémů, slouží jak k evidenci, tak k efektivnímu zpracování a údržbě dat. Díky databázím můžeme snáze získávat z dat informace, jež jsou v dnešní době nepostradatelné při jakékoli lidské činnosti. Pojem „databáze" je všeobecně rozšířen, téměř každý podnik již má vlastní databázi, avšak často se stává, že lidé nerozlišují základní pojmy související s touto problematikou a vše jednoduše nazývají databází. Proto je třeba nejprve definovat základní pojmy týkající se problematiky databází.

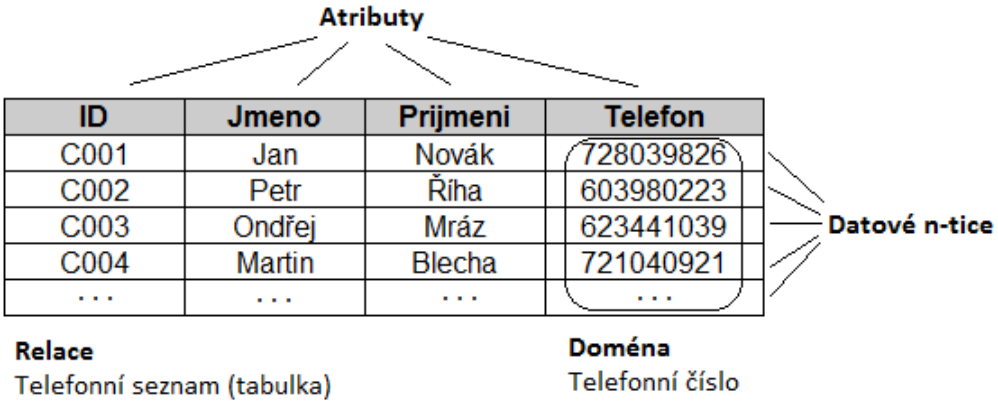
- Systém řízení báze dat (SŘBD) je páteří každého databázového systému, stará se o správný chod databáze a interaguje s uživateli či aplikacemi. Hlavní úlohou SŘBD je zprostředkování řízeného přístupu k databázi, a tak je jakousi membránou mezi uživateli a fyzickými daty. Díky tomu, že přístup k databázím zprostředkovává výhradně SŘBD, můžeme při dobré znalosti relačního modelu vytvářet velice efektivní zabezpečení.[2]
- Databáze (DB) neboli datová základna, jak je často databáze nazývána, je uspořádaná množina dat, která je uložena na paměťovém médiu. Jedná se tedy o data, se kterými pracuje SŘBD, nikoli o konkrétní programy.[9]
- Databázový systém (DBS) je kombinace databáze a SŘBD. Jinak řečeno jedná se o uspořádanou množinu dat, která je řízena systémem řízení báze dat.[7]
- Databázová aplikace je program usnadňující běžnému uživateli přístup k databázi tím, že komunikuje se SŘBD, interpretuje uživatelské požadavky a výsledná data prezentuje v uživatelsky přívětivém rozhraní. Díky databázovým aplikacím není nutné, aby uživatel znal SQL a dokonce ani strukturu databáze samotné. Tyto aplikace jsou programovány ve vyšších programovacích jazycích jako například C++, Java, C#, mohou však být také součástí webového rozhraní.[2]

Z předešlých definic je možné odvodit základní myšlenku databázového zpracování, tou je snaha o oddělení uživatele od přímého kontaktu s fyzickými daty, díky čemuž lze vytvářet velice bezpečné systémy. Nežli se však ponoříme do samotné problematiky zabezpečení je nutné objasnit, co je to relační datový model, a které jeho součásti ho činí tak robustním a bezpečným, díky čemuž jsou dnes relační databáze nejpoužívanější technologií pro uchovávání a práci s daty.

4.1 Relační datový model

Datový model je soubor konceptů popisujících data, vztahy mezi nimi a také omezení, jež se na data vztahují. Jako celek pak model reprezentuje objekty existující v reálném světě. Relační datový model je založen na matematickém principu relací, ty jsou navenek prezentovány jako relační tabulky. Mezi další významné pojmy v relačním datovém modelu patří též atributy, datové n-tice, domény a v neposlední řadě relační databáze. Tyto základní složky relačně datového modelu graficky reprezentuje obrázek 4.1.[2]

- Relace je matematické vyjádření vztahu mezi dvěma objekty, tento vztah je v relačním datovém modelu reprezentován relační tabulkou.
- Atribut je vlastnost objektu z reálného světa, v relačním datovém modelu je reprezentován sloupcem relační tabulky.
- Datová n-tice reprezentuje objekt z reálného světa, který je popsán jednotlivými atributy, v relačním datovém modelu je reprezentován řádkem relační tabulky.
- Doména je množina hodnot přípustných pro nějaký atribut nebo množinu atributů. Domény je třeba nastavit tak, aby atributy odpovídaly skutečným vlastnostem objektů z reálného světa.
- Relační databáze je množina normalizovaných relačních tabulek, které mezi sebou mají určité námi definované vztahy.[8]



Obrázek 4.1: Struktura relační tabulky
Zdroj: Vlastní zpracování.

4.1.1 Vlastnosti relačních tabulek

Na obrázku 4.1 můžeme vidět příklad struktury relační tabulky, pro správné pochopení relačního datového modelu je však zapotřebí pochopit nejen strukturu relačních tabulek, ale také jejich vlastnosti.

- Každá relační tabulka má v rámci databáze unikátní jméno, díky kterému k ní můžeme přistupovat.
- Každá buňka v relační tabulce musí být atomická, což znamená že obsahuje právě jednu hodnotu. Není tedy žádoucí, aby buňka obsahovala například více emailových adres a podobně.
- Každý sloupec relační tabulky musí mít unikátní jméno v rámci tabulky, jestliže pak v různých tabulkách existují atributy se stejným jménem, jejich rozlišení se provádí za pomoci tečkové konvence.
- Veškeré hodnoty v konkrétním sloupci relační tabulky musí odpovídat hodnotám vymezeným doménou, jež k danému sloupci náleží.
- Pořadí sloupců je nevýznamné.
- Každý řádek relační tabulky musí být unikátní, v relační tabulce se tedy nesmí objevit duplicitní záznam.[2]

4.1.2 Klíče relací

Každý záznam relační tabulky musí být unikátní, tato skutečnost má významný vliv na redukci redundance v databázích, jelikož neumožňuje vložení totožných záznamů, a to ani zapříčiněním lidské nepozornosti. Zmíněná vlastnost relačních tabulek je využívána zejména proto, aby bylo možné přistupovat konkrétně k jednotlivým záznamům a jejím atributům. Abychom nemuseli vždy identifikovat záznam pomocí kombinace všech atributů, používají se takzvané klíčové atributy.

K identifikaci záznamu ve vlastní relační tabulce se využívají primární klíče (primary key), ty musí splňovat dva základní požadavky. Primární klíč musí být unikátní, jinými slovy je zapotřebí, aby bylo možné podle primárního klíče vyhledat jeden konkrétní záznam. Primární klíč by také měl být neredukovatelný, mělo by se tedy jednat o nejmenší možnou část záznamu, která stále splňuje požadavek na unikátnost. Potenciálních primárních klíčů může být

v jedné tabulce více, říká se jim kandidátní klíče. Z těchto kandidátních klíčů je třeba vybrat právě jeden, jenž bude co nejlépe reprezentovat danou tabulku a stane se jejím primárním klíčem.[3]

Primární klíče se dají dělit podle způsobu, jakým vznikají, může se jednat o klíče přirozené nebo umělé. Přirozené primární klíče pochází z atributů, které jsou přirozeně součástí tabulky, jako příklady možných přirozených klíčů lze uvést rodné číslo, číslo občanského průkazu nebo pasu a podobně, je také možné využít různé kombinace atributů. Umělé primární klíče jsou, na rozdíl od přirozených, hodnotami, jež do tabulky v jejím přirozeném stavu nespadají a jsou přidány pouze pro účel identifikace řádků tabulky. Využití umělých klíčů sice určitým způsobem zvýší objem dat, jež bude v databázi uchováván, avšak přináší vyšší úroveň zabezpečení, jelikož dále není nutné vyhledávat záznamy za pomoci atributů, které by mohly být citlivými daty, jako je tomu v případě využití přirozených primárních klíčů. Díky této vlastnosti jsou ve firemních databázích dnes využívány především umělé primární klíče.[2] Avšak i při nasazení umělých primárních klíčů vzniká dilema, jež staví návrháře databází proti sobě. První skupina se přiklání k využití datového typu NUMBER, jelikož je rychlejší a navíc díky možnosti auto inkrementace i snáze využitelný. Druhá skupina se naopak přiklání k využití datového typu CHAR, ten sice není tak rychlý jako NUMBER, avšak může nést aditivní informace o daném objektu. Tento spor samozřejmě nemá jasného vítěze, protože pro každý typ entity může být lépe využitelný jiný typ primárního klíče, osobně se však přikláním spíše k využití typu CHAR, hlavním důvodem je úspora množství dat v databázi, jelikož primární klíč typu CHAR může často nahradit i několik atributů, je-li správně navržen.

Dalším druhem klíčových hodnot jsou takzvané cizí klíče (foreign key), díky nimž lze vytvářet vztahy 1:N, toho je dosaženo vložením primárního klíče nadřazené tabulky do tabulky podřízené, ten se následně stane v podřízené tabulce cizím klíčem. Jelikož cizí klíče jsou primárními klíči v nadřazených tabulkách je nutné, aby splňovaly stejná kritéria jako běžné primární klíče.[3]

4.1.3 Relační integrita

Nyní, když víme jaké prvky relační model obsahuje a v jakém jsou navzájem vztahu, je třeba vysvětlit jakým způsobem relačně databázový systém zajišťuje, aby veškerá data v relační databázi odpovídala pravidlům pro relační tabulky, struktuře dané evidence a také skutečným vlastnostem reálných objektů. Nejprve je třeba vytvořit soubor pravidel, která zajistí kontrolu při vkládání dat do databáze a jejich modifikaci, aby nedošlo k poškození či znehodnocení databáze. Těmto pravidlům se odborně říká integritní omezení, musí být vytvořena již při

návrhu databáze, a jejich implementace probíhá ve fázi konstrukce jednotlivých relačních tabulek. Integritní omezení se dělí na tři základní druhy.[8]

- Entitní integrita zajišťuje, aby každý záznam v relační tabulce byl unikátní. Toho je dosaženo takovým způsobem, že každá tabulka musí mít definovaný primární klíč. Primární klíč musí být vyplněn pro každý záznam a jeho hodnota musí být v dané tabulce unikátní.
- Referenční integrita zajišťuje korektnost vztahů definovaných schématem databáze. Aby byly tyto vztahy realizovány správným způsobem, je třeba zajistit, že hodnota cizího klíče, jestliže v tabulce nějaký existuje, bude přiřaditelná k hodnotě existujícího záznamu nadřazené tabulky nebo zůstane nevyplněná, to záleží na konkrétním nastavení integritních omezení.
- Doménová integrita se stará o to, aby veškeré hodnoty ve sloupci odpovídaly předem definovaným parametrům pro konkrétní atribut. Díky doménové integritě pak není možné záznam uložit, jestliže kterákoli z hodnot není prvkem domény definované pro konkrétní sloupec.[9]

4.1.4 Relační jazyky

Aby bylo možné interagovat se SŘBD, je nejprve třeba definovat formální prostředky, jež by umožnily definovat jednotlivé prvky relačního datového modelu a také, aby bylo možné provádět nad těmito prvky jednotlivé operace. E. F. Codd navrhl pro relační datový model v roce 1972 dva jazyky, které byly k tomuto účelu využívány před příchodem SQL, to je jakousi nadstavbou těchto jazyků a umožňuje kromě provádění základních operací i využití různých pokročilých funkcí. Těmito jazyky byly relační algebra a relační kalkul. Relační algebra je procedurálním jazykem, naopak relační kalkul je jazykem neprocedurálním, avšak oba tyto jazyky jsou navzájem převoditelné. Z relační algebry a kalkulu vychází dnes používané relačně databázové jazyky vyšší úrovně jako jsou například SQL či QBE.[2]

SQL neboli česky strukturovaný dotazovací jazyk je dnes primárním jazykem relačních databází. Tento dotazovací jazyk se objevil již v roce 1974, kdy byl ještě nazýván SEQUEL. V roce 1986 se SQL stalo standardem využívaným ve všech databázových systémech až dodnes. Standardizaci má na starosti organizace ANSI. Ta původně chtěla rozvinout zcela nový dotazovací jazyk, avšak SQL bylo již v době, kdy se zrodila myšlenka standardního dotazovacího jazyka, poměrně rozšířené a dobře se osvědčilo, proto byl pro standard nakonec využit právě tento dotazovací jazyk.[7] Nejnovějším standardem pro SQL je SQL: 2011.

SQL je deklarativním, relačně úplným, dotazovacím jazykem. Deklarativní znamená, že neurčujeme, jak něco provést ale pouze, čeho je třeba dosáhnout. Dotazovací proto, že je vytvořen pro komunikaci se SŘBD. A relačně úplný pak znamená, že v něm lze vytvořit libovolný dotaz, jenž je možné zkonstruovat v relačním kalkulu či algebře, k tomu se využívá především základní příkaz jazyka, SELECT, jenž pracuje na principech selekce, projekce a spojení. Aby SQL splňovalo požadavky kladené na dotazovací jazyk, je třeba aby umožňoval definici dat, integritních omezení, manipulaci s daty, práci s transakcemi, vytváření funkcí, pohledů, bezpečnostních prvků umožňujících autorizaci a další. Pro každou z těchto činností využíváme některou ze tří složek jazyka SQL.[9] Při vytváření efektivních bezpečnostních opatření pak využíváme dokonce každou z nich. SQL se tedy skládá z:

• Data Definition Language (DDL), neboli jazyk pro definici dat. Tato složka jazyka SQL umožňuje vytvářet, upravovat či mazat databázové struktury jako jsou tabulky, indexy, pohledy a další objekty. Mezi DDL příkazy patří: CREATE, ALTER a DROP.

• Data Manipulation Language (DML), neboli jazyk pro manipulaci s daty. Tato složka jazyka SQL umožňuje získávání dat z databáze a jejich úpravy. Mezi DML příkazy patří: SELECT, INSERT a UPDATE.

• Data Control Language (DCL), neboli jazyk pro řízení dat. Tato složka jazyka SQL umožňuje nastavování přístupových práv a řízení transakcí. Mezi DCL příkazy patří: GRANT, REVOKE, COMMIT, ROLLBACK, START TRANSACTION a další.[5]

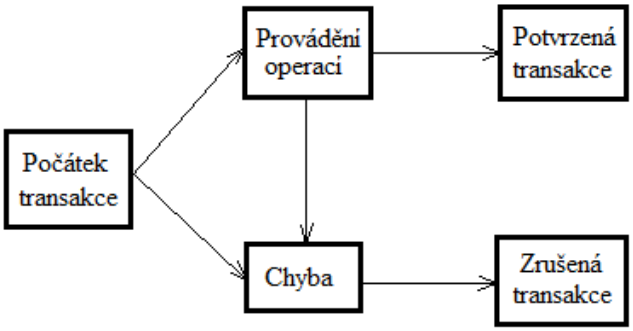
Zásluhou veškerých možností, jež nám nabízí, je SQL velice silným nástrojem, s jehož pomocí můžeme jednoduše vytvářet robustní a dobře zabezpečené databáze, navíc díky standardizaci jsou takto vytvořené databáze nezávislé na konkrétní implementaci databázového systému.

4.2 Transakční zpracování

Transakční zpracování je charakteristickým rysem relačních databázových systémů, zajišťuje bezpečný a efektivní chod databáze. Největším přínosem je pak transakční zpracování v případech, kdy je nutné aby se stejnými daty pracovalo více uživatelů ve stejný okamžik. Základem transakčního zpracování jsou jednotlivé transakce, ty se skládají ze série operací pro manipulaci s daty. Jednotlivé operace jsou zapsány za pomoci již zmíněného jazyka SQL. Aby bylo možné transakci úspěšně dokončit a vytvořit tak nový perzistentní stav databáze, je třeba ji potvrdit, k tomuto účelu se v jazyce SQL využívá příkaz COMMIT. K potvrzenému stavu se lze následně kdykoli, v případě potřeby, vracet za pomoci příkazu ROLLBACK.[7]

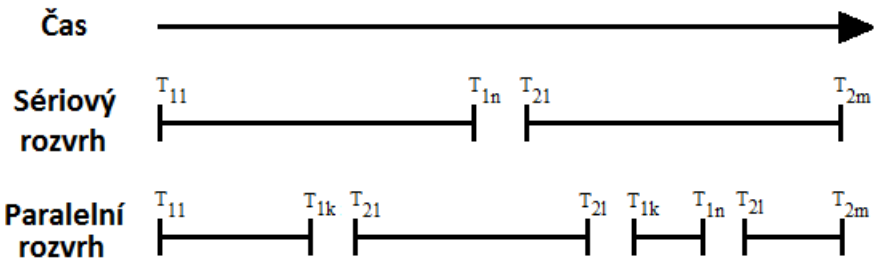
Transakce musí mít několik základních vlastností. Především musí být atomické, a tedy buď proběhnou celé a nebo vůbec. Také musí být konzistentní, což znamená, že při provádění transakce není porušeno žádné z integritních omezení. Jestliže by nastala chyba a hrozilo by uvedení databáze do nekonzistentního stavu, takováto transakce by nebyla dokončena a došlo by k automatickému vyvolání příkazu ROLLBACK, jenž by byl iniciován službami SŘBD pro zotavení systému. Každá transakce dále musí být nezávislá, a tedy dílčí změny konkrétní transakce nejsou ostatním transakcím přístupné, dokud nedojde k dokončení původní transakce. Poslední podmínkou je pak trvalost. Význam této vlastnosti spočívá v tom, že jakmile je transakce jednou ukončena, změny provedené na databázi jsou již trvalé a databáze je tak převedena do nového perzistentního stavu. Zmíněné vlastnosti bývají označovány pojmem ACID, tato zkratka pochází z anglických názvů jednotlivých vlastností.[5] Každá transakce prochází životním cyklem, ten může zahrnovat následující stavy, do nichž se transakce během životního cyklu může dostat. Jednotlivé stavy pak jsou zobrazeny obrázkem 4.2.

- Počátek transakce - stav, ve kterém se transakce nachází při svém vzniku.
- Provádění operací - stav, ve kterém se nachází transakce ve chvíli, kdy jsou prováděny operace v ní obsažené.
- Potvrzená transakce - stav, ve kterém se nachází transakce, je-li úspěšně ukončena a potvrzena příkazem COMMIT.
- Chyba - stav, ve kterém se nachází transakce, dojde-li k nějaké chybě před jejím ukončením.
- Zrušená transakce - stav, ve kterém se nachází transakce poté, co se provede operace ROLLBACK.[7]



Obrázek 4.2: Životní cyklus transakce
Zdroj: Vlastní zpracování podle Pokorný, Valenta, 2013.[7]

Transakcí jakožto logických jednotek transakčního zpracování může být samozřejmě více, což je způsobeno větším počtem uživatelů pracujících s databází ve stejný okamžik. Aby nedocházelo k chybám způsobeným chaotickým prováděním operací, je nutné aby SŘBD uměl správně plánovat provádění jednotlivých operací. O vytváření rozvrhů, podle kterých se jednotlivé operace provádí, se stará plánovač, ten je součástí SŘBD. Je možné vytvářet sériové rozvrhy, v takovém případě jsou uskutečňovány transakce postupně za sebou. Další možností je pak paralelní zpracování transakcí, to je sice náročnější na výpočetní výkon, avšak poskytuje vyšší výkon díky efektivnějšímu řazení operací. Na obrázku 4.3 můžeme vidět příklad sériového a paralelního rozvrhu dvou transakcí T_1 a T_2, které jsou složeny z operací $\{T_{11}, ..., T_{1n}\}$ a $\{T_{21}, ..., T_{2m}\}$.[7]



Obrázek 4.3: Příklady rozvrhů pro provedení transakcí
Zdroj: Vlastní zpracování

Jak může být patrné z předchozích částí, relačně databázová technologie přináší mnoho výhod. Mezi nejvýznamnější z nich patří efektivní prostředky bránící vzniku redundance dat a metody chránící relační databáze před uvedením do nekonzistentního stavu. Další výhodou je omezení rizikovosti souběžné práce více uživatelů či aplikací se stejnými daty. V neposlední řadě jde též o vysokou míru datové integrity, která nám umožňuje tvořit a SŘBD vynucovat pravidla integritních omezení. Je nutné zmínit také vzájemnou nezávislost dat a aplikací, jež zvyšuje životnost aplikací, usnadňuje přenositelnost dat a napomáhá tak k vyšší flexibilitě databázových systémů. Díky těmto a dalším výhodám jsou relačně databázové evidence mnohem lépe chráněny před různými nebezpečími, jež by mohla ohrozit relevantnost dat v nich uložených. Přesto, že relačně databázový přístup s sebou, kromě výhod, nese také některé nevýhody, jeho využití je dnes velmi rozšířené. Dalo by se říci, že relačně databázová technologie dnes tvoří páteř většiny systému, ve kterých je třeba zpracovávat velké objemy dat. To je způsobeno především vlastnostmi relačně databázové technologie, které jako celek tvoří stabilní základnu, na níž je možné dále tvořit efektivní zabezpečení. Nástroje, jež jsou k tomuto účelu využívány, jsou popsány v následující kapitole.

5 Zabezpečení relačně databázových systémů

V této kapitole se zaměříme na jednu z klíčových úloh při vytváření a správě databází, na zabezpečení. Nejprve se podíváme na to, proč je nevhodné brát zabezpečení na lehkou váhu a jaká potenciální nebezpečí databázi mohou ohrozit. Dále si popíšeme některé z mechanismů, využívaných k zabezpečení databáze. Praktické využití popisovaných mechanismů v rámci zabezpečení konkrétní databáze je pak možné nalézt v šesté kapitole bakalářské práce, ta je věnována praktickým ukázkám.

V dnešní době, kdy informace jsou jedním ze základních stavebních kamenů konkurenceschopnosti firmy, jsou velice důležité nejen metody získávání informací, ale rovněž je třeba zajistit jejich ochranu. Informace jsou dnes ukládány do velké míry právě v relačních databázích a jejich ztráta, poškození, zneužití nebo nedostupnost by mohla mít katastrofální následky, proto je třeba se zabývat problematikou zabezpečení databázových systémů velice podrobně, především pak u podnikatelských subjektů, v jejichž databázích se velmi často vyskytují informace strategického významu. Zabezpečení databázového systému však nezahrnuje pouze data uchovávaná v databázích, je třeba brát v potaz i veškeré faktory jako jsou hardware, software, data a lidé.[2]

5.1 Bezpečnostní rizika a jejich možné následky

Ještě před tím, než se budeme moci hlouběji ponořit do problematiky zabezpečení databázových systémů, je třeba si uvědomit jaká rizika společnosti hrozí a jaké následky z nich mohou plynout, jestliže nebudou správně nastavena bezpečnostní opatření. Následky narušení bezpečnosti lze rozdělit do následujících pěti kategorií:

- Krádež a zpronevěra - tato kategorie je velmi závažná a má dopad spíše na organizaci jako celek, než na databázi. Může se jednat o různé typy napadení, nicméně krádež či zpronevěra je vždy způsobena záměrnou lidskou činností, mající za cíl obohacení se na úkor organizace. Při krádeži a zpronevěře nemusí nutně dojít ke změně dat, což platí také pro činnosti vedoucí ke ztrátě utajení či soukromí.
- Ztráta utajení - předmětem utajení jsou především data strategického charakteru, jako jsou například různé postupy či parametry výroby a podobně. Velké množství podnikatelských subjektů uchovává ve svých databázích data, jež jsou kriticky důležitá pro jejich podnikatelskou činnost, ta by měla být zpřístupněna pouze autorizovaným osobám. Ztráta utajení může mít za následek snížení konkurenceschopnosti organizace.

- Ztráta soukromí - soukromí se vztahuje především na citlivá data osob, o nichž je v databázi veden záznam. Většina podnikatelských subjektů vede ve svých databázích záznamy o zaměstnancích či klientech, jejichž součástí jsou citlivá data a stejně jako u strategických dat podléhajících utajení, i k těmto datům by měly mít přístup pouze autorizované osoby. Ochrana citlivých dat je garantována zákonem a ztráta soukromí může mít za následek právní kroky proti organizaci.
- Ztráta integrity - jak již bylo zmíněno v části 4.1.3 integrita je důležitá pro správný chod databáze. A je proto třeba zajistit, aby nedošlo k jejímu narušení. Jestliže databáze ztratí integritu dat, může to vést k neplatnosti nebo znehodnocení dat, a to může vážně ovlivnit veškeré činnosti organizace.
- Ztráta dostupnosti - dostupnost je důležitá zejména pro organizace poskytující nepřetržité virtuální služby, může se jednat například o internetové obchody, poskytovatele server hostingu a podobně. Jestliže dojde ke ztrátě dostupnosti, může to mít za následek značné finanční ztráty společnosti.[2]

Lze si povšimnout, že při zanedbání bezpečnosti se následky projeví nejen na databázi samotné, ale ovlivní též chod organizace jako celku. Ať již se jedná o ztráty hmotné, jako například data, hardware, software či nehmotné, jako jsou ztráta důvěry zákazníků či konkurenceschopnosti firmy, pro podnikatelské subjekty jsou nežádoucí. V tabulce 1 jsou zobrazeny příklady potenciálních bezpečnostních rizik. Jak můžeme vidět různá rizika mohou mít odlišné následky, které se mohou projevit jak na databázi tak i na firmě samotné. Ke každému typu ohrožení je tedy nutné přistupovat jiným způsobem. Proto v dnešní době firmy vynakládají poměrně velké množství času a finančních prostředků na vytváření co možná nejbezpečnějších systémů.

Každé z bezpečnostních rizik uvedených v tabulce 1 má svého původce, může se jednat o zaměstnance společnosti, osoby mimo společnost, defektní hardware či chybný software a v neposlední řadě též špatně navržená bezpečnostní opatření. Proto je třeba si při návrhu zabezpečení uvědomit nejen jaká rizika mohou databázi reálně hrozit, ale též odkud by mohlo dojít k jejímu narušení. Může se jednat o narušení bezpečnosti interní či externí. Jestliže však správně vystihneme podstatu bezpečnostních rizik, je mnohem snazší navrhnout bezpečný systém.

V rámci návrhu bezpečnostního protokolu společnosti je třeba vzít v potaz tři základní typy zabezpečení. Jedná se o zabezpečení fyzické, síťové a zabezpečení databáze. Každé z těchto typů zabezpečení napomáhá zabránit různým hrozbám a využívá jiných prostředků.

Tabulka 1: Příklady možných rizik a jejich následků

Potenciální bezpečnostní rizika	Krádež a zpronevěra	Ztráta utajení	Ztráta soukromí	Ztráta integrity	Ztráta dostupnosti
Použití přístupových prostředků jiné osoby	X	X	X		
Neautorizovaná úprava nebo kopírování dat	X	X	X		
Pozměnění programu	X			X	X
Zachycování přenášených zpráv	X	X	X		
Průnik hackera	X	X	X		
Krádež dat, programů či vybavení	X	X	X		X
Neodpovídající školení zaměstnanců	X	X	X	X	X
Prohlížení a prozrazení neautorizovaných dat	X	X	X		
Elektronické rušení a vyzařování				X	X
Poškození dat kvůli přerušení dodávky energie				X	X
Fyzické poškození zařízení				X	X
Zhroucení systému SŘBD či operačního systému				X	X
Působení počítačových virů				X	X

Zdroj: Conolly, 2009.[2]

- Fyzické zabezpečení zahrnuje především opatření zabraňující fyzickému vniknutí do objektu či opuštění objektu s majetkem společnosti. Fyzické zabezpečení zabraňuje především zcizení a poškození hardwaru, softwaru nebo dat. Může se jednat o různé kamerové systémy, bezpečnostní zámky nebo fyzickou ostrahu a podobně.

- Síťové zabezpečení je zapotřebí především, je-li databáze přístupná přes internet, je tedy nezbytností pro podnikatelské subjekty, jež umožňují přístup ke své databázi prostřednictvím webových stránek jako jsou například banky, e-shopy a podobně. Hlavním cílem síťového zabezpečení je zajistit, aby se k datům nedostaly neautorizované osoby, popřípadě aby nedošlo k poškození databáze nebo dat v ní uchovávaných.

Jako nástroje síťového zabezpečení lze uvést firewally, proxy servery, filtrování paketů a další.

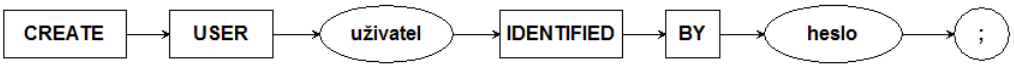
- Zabezpečení databáze spočívá především v zajištění toho, aby každý z uživatelů měl přístup pouze k datům, ke kterým je oprávněn přistupovat, čehož je dosahováno nejčastěji za použití mechanismů jako jsou pohledy, oprávnění, role a v neposlední řadě též uživatelské účty a profily. Dále je třeba zajistit správné fungování databáze, aby se dokázala vyrovnat i s neočekávanými chybami, k tomu se využívají především nativní funkce SŘBD jako například transakce, služby zotavení, kontrola integrity, řízení paralelního přístupu a další, viz kapitola 4.[5]

Je možné si povšimnout, že různé typy zabezpečení se do jisté míry překrývají a směřují ke stejnému cíli, kterým je snaha o vytvoření co možná nejbezpečnějšího prostředí. Při tvorbě komplexního, firemního systému zabezpečení je většinou nutné využít veškeré již zmíněné typy zabezpečení. V této bakalářské práci však bude nadále popisováno pouze zabezpečení databáze samotné, a to především hlavní nástroje k tomu využívané, tedy pohledy a mechanismy sloužící k autorizaci, mezi něž patří uživatelské účty, profily, oprávnění a role. Každý z uvedených nástrojů má v rámci bezpečnosti jinou úlohu, liší se však nejen zaměřením, ale především výsledným efektem a také mírou znalostí, jež jsou nutné k jejich realizaci.

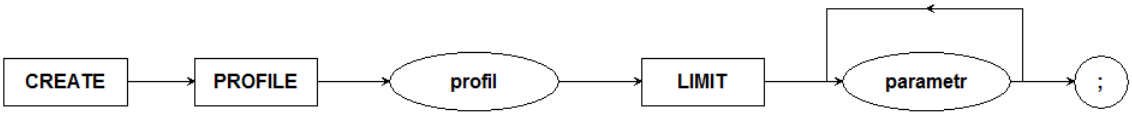
5.2 Autorizační mechanismy

V podnikové sféře mívají databáze většinou velké množství uživatelů. Aby bylo možné účinně databázi chránit před poškozením právě ze strany uživatelů, ať již úmyslným či nikoli, je třeba, aby databázový systém poskytoval nástroje zamezující neoprávněným vstupům. Takováto potřeba vzniká zejména z důvodu utajení důležitých dat, jako jsou například citlivé údaje a informace strategického významu, ke kterým by měly mít přístup pouze autorizované osoby. Právě z důvodu utajování některých citlivých a strategických informací jsou autorizační mechanismy dnes jedny z nejdůležitějších a nejvyužívanějších prostředků k zabezpečení databáze. Díky nim může ve firemních databázích vznikat bezpečnostní hierarchie s různými úrovněmi autorizace založená na skutečné hierarchii podniku. To v běžné praxi znamená, že v rámci bezpečnostních opatření se udělují pouze taková oprávnění, jež jsou nezbytná pro práci konkrétního pracovníka, o nic víc ani méně.

Mezi mechanismy sloužící k autorizaci se v relačně databázových evidencích řadí uživatelské účty a profily, jež definují bezpečnostní prvky uživatelského účtu, dále pak oprávnění a role. Tyto nástroje bývají velmi často kombinovány například s pohledy, aby vytvořili komplexní ochranu před zneužitím informací či poškozením databáze ze strany uživatelů.



Obrázek 5.1: Struktura SQL příkazu CREATE USER
Zdroj: Vlastní zpracování.



Obrázek 5.2: Struktura SQL příkazu CREATE PROFILE
Zdroj: Vlastní zpracování.

5.2.1 Uživatelské účty a profily

Každý podnikatelský subjekt, ať již větší či menší, zaměstnává nějaké pracovníky, z nichž většinou jistá část k výkonu své práce vyžaduje přístup k datům, jež jsou uložena ve firemních databázích. Aby se však pracovník k požadovaným datům mohl dostat, je nejprve nutné mu vytvořit uživatelský účet, s jehož pomocí se bude moci k databázi přihlásit. Vytváření uživatelských účtů se provádí za pomoci SQL příkazu CREATE USER. Základní struktura příkazu CREATE USER je graficky znázorněna na obrázku 5.1.

Každý uživatel je definován svým přihlašovacím jménem a heslem. Heslo je defaultně kontrolováno přímo v databázovém systému, existuje však možnost využití namísto klauzule IDENTIFIED BY alternativu v podobě IDENTIFIED EXTERNALLY, jestliže se rozhodneme pro tuto možnost, pak je identita uživatele kontrolována externě, například operačním systémem nebo nějakým jiným programem.[4] Možnost IDENTIFIED EXTERNALLY bych však osobně doporučil využívat výhradně u operačních systémů, jež mají dostatečně silné zabezpečení, nejlépe tedy u operačních systémů na bázi UNIXu.

Heslo však není jediným parametrem, který lze v rámci uživatelského účtu modifikovat. Pro zvýšení bezpečnosti se velmi často k uživatelskému účtu připojují takzvané profily, ty umožňují modifikaci nejrůznějších bezpečnostních parametrů uživatelského účtu. Profily je možné definovat SQL příkazem CREATE PROFILE, jehož základní struktura je graficky znázorněna na obrázku 5.2. Mezi nejdůležitější parametry v rámci zabezpečení, které je možné za pomoci profilů modifikovat, pak patří následující:

- SESSIONS_PER_USER - Za pomoci tohoto parametru lze nastavit množství relací, které může mít uživatel v jeden okamžik aktivní.

- CONNECT_TIME a IDLE_TIME - Těmito parametry lze nastavit maximální doba, po kterou může být uživatel přihlášen v případe CONNECT_TIME a jak dlouho může uživatel setrvat v nečinnosti v případě IDLE_TIME. Po uplynutí této nastavené doby bude uživatel automaticky odhlášen.

- FAILED_LOGIN_ATTEMPTS - Za pomoci tohoto parametru můžeme nastavit počet neúspěšných pokusů o přihlášení k databázi, po překročení tohoto limitu bude účet dočasně uzamčen. Uživatelský účet bude následně odemčen po uplynutí doby nastavené parametrem PASSWORD_LOCK_TIME, popřípadě může být ručně odemčen administrátorem.

- PASSWORD_LIFE_TIME - Za pomoci tohoto parametru lze nastavit délku platnosti hesla, po jejímž uplynutí bude třeba heslo změnit, jinak vyprší jeho platnost a dále nebude možné se k databázi přihlásit. Díky PASSWORD_GRACE_TIME je pak možné nastavit časové období, ve kterém bude uživatel upozorňován na potřebu změny hesla, než dojde k jeho konečné expiraci.

- PASSWORD_REUSE_TIME a PASSWORD_REUSE_MAX - Tyto parametry umožňují zabránit opakovanému využívání stejných hesel a donutit tak uživatele hesla doopravdy měnit. PASSWORD_REUSE_TIME zabraňuje použití stejného hesla po určitou dobu, zatímco PASSWORD_REUSE_MAX nastavuje počet změn hesla, jež musí minimálně proběhnout, než bude možné opět využít stejné heslo. Aby měly požadovaný efekt, je vhodné využít tyto parametry současně.

- PASSWORD_VERIFY_FUNCTION - Za pomoci tohoto parametru je možné nastavit verifikační funkci, jež bude dohlížet na to, aby uživatelé nepoužívali příliš triviální hesla. Heslo pak musí odpovídat námi definovaným parametrům. Lze například nastavit minimální délku hesla, povinný počet malých či velkých písmen, číslic nebo aby se jeho části neshodovali s uživatelským jménem a podobně. V databázích Oracle je možné buď využít přednastavené funkce, anebo si vytvořit vlastní podle svých specifických požadavků.[4, 6]

Díky kombinaci výše uvedených parametrů pak můžeme docílit značného snížení rizika spojeného s použitím přístupových prostředků druhé osoby. Například lze díky určení složitosti hesla a doby jeho platnosti znesnadnit útočníkovy získávání hesla. Dále nastavením CONNECT_TIME a IDLE_TIME, snížíme riziko zneužití počítače s již přihlášenou relací, který byl ponechán bez dozoru. A v neposlední řadě pak nastavením FAILED_LOGIN_ATTEMPTS můžeme zabránit opakovaným pokusům útočníka o prolomení hesla například pomocí různých programů či robotů. Při nastavování těchto parametrů je však nutné být velmi opatrný

a brát v potaz nejen bezpečnost systému, ale také uživatele samotné. Jestliže bychom totiž vytvořili příliš tvrdá bezpečnostní opatření, docílili bychom spíše snížení pracovní efektivity uživatelů, což je efekt, kterému je třeba se vyhnout.

Kromě již zmíněných parametrů však existují i další, ty nicméně spíše než k zabezpečení databáze slouží k omezení zdrojů uživatelského účtu. S jejich pomocí je pak možné například ovlivnit jaké množství procesorového času může uživatel v relaci maximálně využít, jaké množství prostoru pro něj bude v rámci databáze vyhrazeno a podobně.

5.2.2 Oprávnění

Po úspěšném vytvoření uživatelského účtu a nastavení potřebných parametrů profilu je dále třeba přidělit k uživatelskému účtu oprávnění, jež definují, k jakým akcím je v rámci databáze uživatel autorizován. Pokud bychom tento krok neprovedli, uživatel by měl pouze oprávnění, jež jsou nastavena v roli PUBLIC, tato oprávnění jsou přidělena automaticky všem uživatelům a nelze je odebrat.[1] V ideálním případě jsou oprávnění uživatelům přidělována ve formě rolí, tento mechanismus je popsán v části 5.2.3, jelikož tím docílíme úspory zdrojového kódu, a tedy i času a finančních prostředků. Nežli však uživateli přidělíme nějaká oprávnění, je na místě důkladně prozkoumat, jakými oprávněními by měl konkrétní uživatel disponovat, abychom databázi nevystavili zbytečnému riziku.

Dnes existují dvě hlavní politiky přidělování práv, z nichž každá má své přednosti i slabiny. První z nich spočívá v postupném přidělování práv s tím, jak se zaměstnanec osvědčí. Výhoda této metody spočívá ve vyšší bezpečnosti, jelikož přidělujeme oprávnění postupně, s tím jak roste zaměstnancova kvalifikace, také si můžeme být jistější jeho loajalitou. Díky tomu by se nemělo stát, že zaměstnanec dostane oprávnění k činnostem, pro něž není kvalifikován, nebo taková, kterých by mohl zneužít. Nevýhodou této metody je pak snížená možnost delegování úkolů na podřízené, jelikož se může stát že zaměstnanec, na kterého by měl být úkol delegován, nemá dostatečná práva přístupu. Dalším problémem je vyšší časová náročnost oproti metodě druhé, neboť při použití této metody je třeba častěji měnit zaměstnancům oprávnění. Druhá metoda je pak pravým opakem metody první, zaměstnancům přidělíme veškerá oprávnění, která na své pozici v rámci společnosti mohou mít, ta pak následně odebíráme až na základě špatné zkušenosti. Tato metoda není tak bezpečná, jelikož zaměstnanci velmi snadno získají oprávnění k činnostem, pro které nemusí být kvalifikovaní a mohou v nich tedy způsobit i vážnější problém, jehož náprava pak může stát velké množství času a peněz. U této metody je také poněkud vyšší pravděpodobnost zneužití pravomocí. Výhodou oproti předchozí metodě je však její podpora delegování úkolů a snazší nahrazení chybějících pracovníků. Ani jedna z těchto metod však není výrazně lepší než druhá, a proto

se dnes v praxi můžeme setkat s každou z nich, nicméně osobně bych se přiklonil spíše k politice postupného přiřazování práv, právě pro její vyšší bezpečnost.

Díky tomu, že umožňují konkrétně specifikovat úroveň autorizace jednotlivým uživatelům, jsou oprávnění velmi důležitým nástrojem k zabezpečení databáze. Přiřazení práv se provádí příkazem GRANT, je také možné práva odebírat příkazem REVOKE. Jestliže u přiřazení oprávnění použijeme klauzuli „WITH ADMIN OPTION", může uživatel přidělovat oprávnění, jež mu byla touto cestou poskytnuta, dalším uživatelům, stejně jako tomu je v případě práv ke strukturám, které sám vlastní. Takovýto postup by ovšem mohl ohrozit bezpečnost, a tak ve firemních databázích využívá pouze zřídka.[4]

Oprávnění se dělí na dvě skupiny, systémová oprávnění a objektová oprávnění, každá skupina ovlivňuje jinou část práce s databází a také mají lehce odlišnou strukturu SQL příkazů GRANT a REVOKE, ty jsou graficky reprezentovány na obrázcích 5.3 a 5.4.

Systémová oprávnění

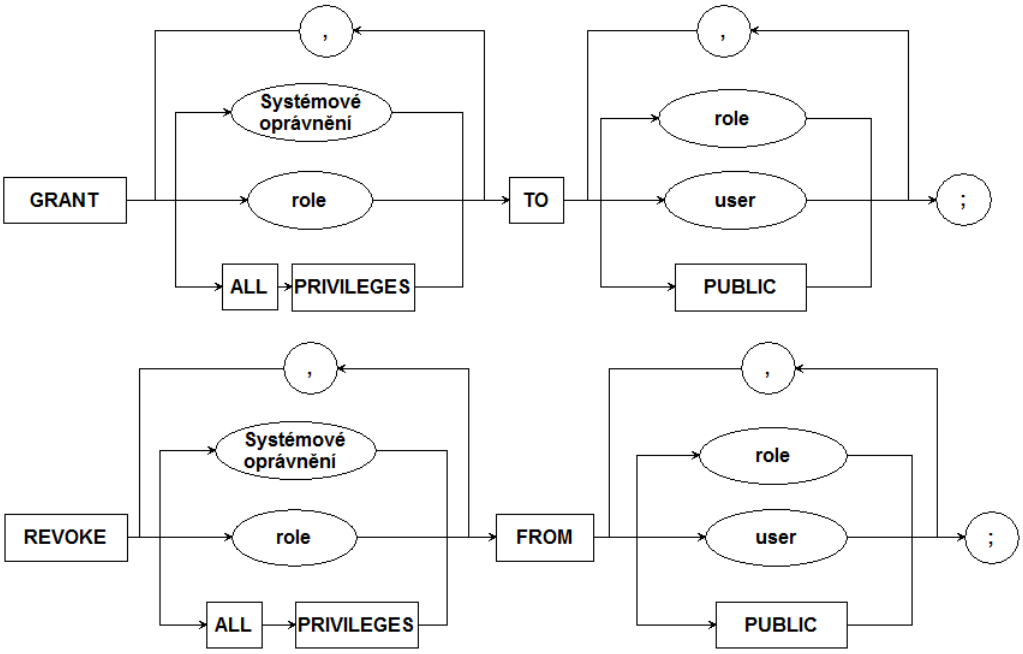
Systémová oprávnění umožňují uživateli provádět specifické sady příkazů nad různými databázovými objekty. Za pomoci systémových oprávnění lze modifikovat strukturu databáze. Proto je tento typ oprávnění přidělován většinou pouze osobám starajícím se o databázi a její vývoj. Systémová oprávnění může přidělit nebo odebrat pouze administrátor databáze nebo uživatel s administrátorskými oprávněními.[6] Mezi systémová oprávnění patří následující:

- CREATE - Umožní uživateli vytvářet v databázi objekty předem specifikovaného typu.
- ALTER - Umožní uživateli měnit v databázi strukturu objektů předem specifikovaného typu.
- DROP - Umožní uživateli mazat z databáze objekty předem specifikovaného typu.[1]

Konkrétní typy objektů, se kterými může uživatel za pomoci systémových oprávnění pracovat, jsou například tabulky, pohledy, materializované pohledy, indexy, sekvence, PL/SQL funkce, procedury a další.[6]

Objektová oprávnění

Objektová oprávnění umožňují provádět specifické sady příkazů nad již existujícími databázovými objekty. Mezi tyto objekty patří tabulky, pohledy, materializované pohledy, indexy, sekvence, PL/SQL funkce, procedury a další, stejně jako je tomu v případě systémových



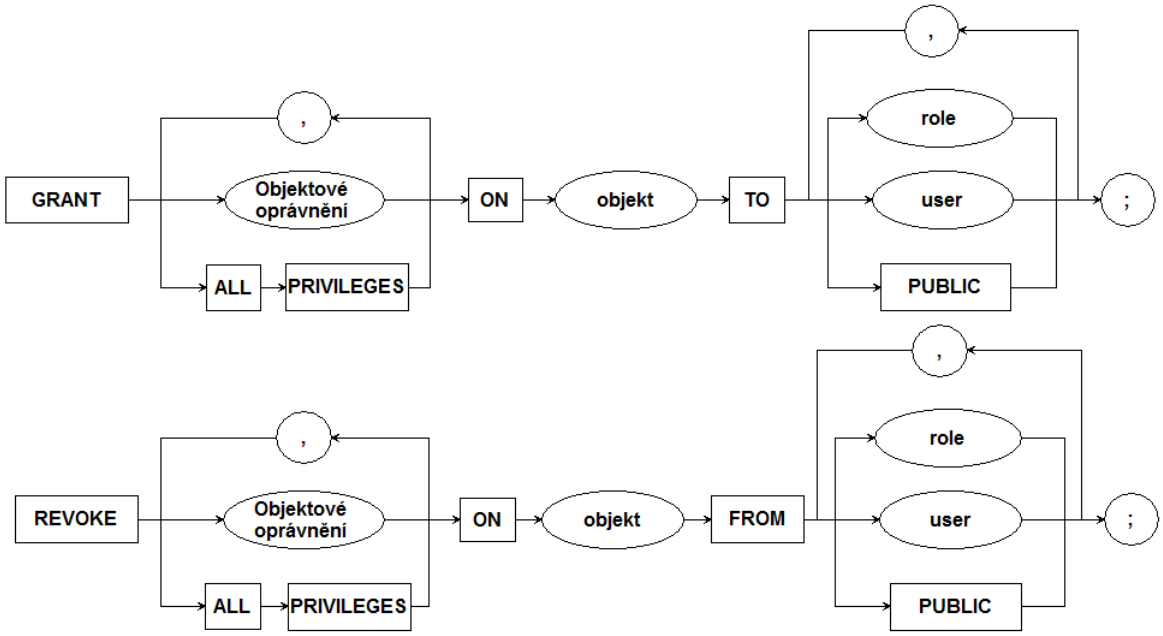
Obrázek 5.3: Struktura SQL příkazů GRANT a REVOKE pro systémová oprávnění
Zdroj: Vlastní zpracování.

oprávnění. Rozdíl je však v tom, že vlastník objektu disponuje všemi objektovými oprávněními k danému objektu a tyto práva mu nemohou být odebrána, dokud je vlastníkem tohoto objektu. Jakožto vlastník objektu má také uživatel právo přidělovat objektová oprávnění ke svým objektům jiným uživatelům, těm však již mohou být tato oprávnění odebrána.[6] Mezi objektová oprávnění patří následující:

- EXECUTE - Umožní uživateli spouštět PL/SQL funkce či procedury.
- SELECT - Umožní uživateli vyhledávat v tabulkách, pohledech, materializovaných pohledech nebo sekvencích.
- INSERT - Umožní uživateli vkládat data do tabulek.
- UPDATE - Umožní uživateli aktualizovat data v tabulkách.
- DELETE - Umožní uživateli odebírat data z tabulek.[1]

5.2.3 Role

Mechanismus rolí slouží v databázové technologii ke sdružování práv. Jedná se v podstatě o pojmenovanou skupinu oprávnění, kterou je možné jako celek přiřadit libovolnému uživateli. Kromě samostatných oprávnění můžeme k roli přiřazovat také další role. Díky těmto



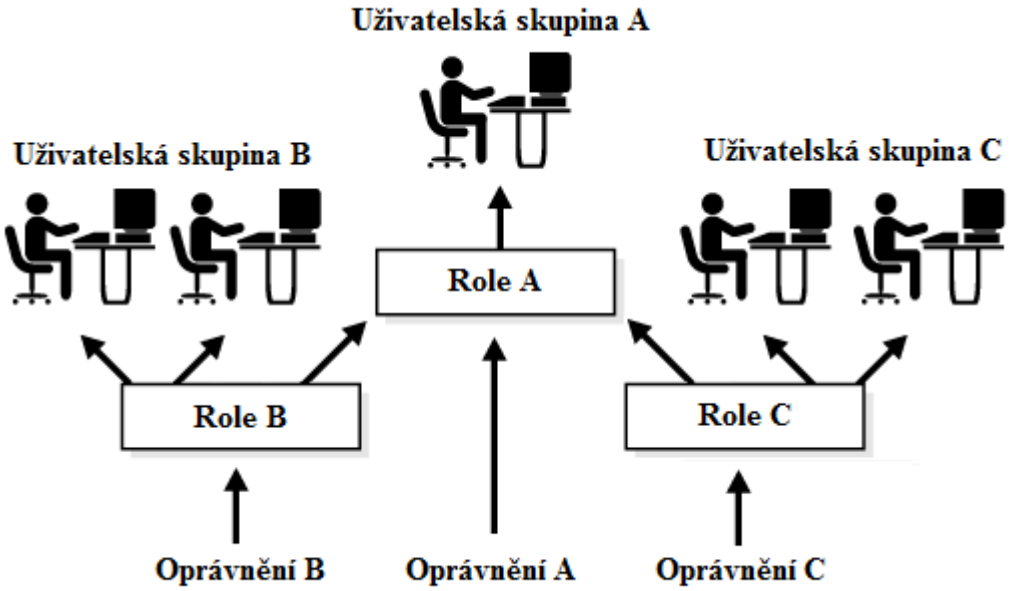
Obrázek 5.4: Struktura SQL příkazů GRANT a REVOKE pro objektová oprávnění
Zdroj: Vlastní zpracování.

skutečnostem je možné vytvořit přehlednou hierarchickou strukturu oprávnění, jež budou sdružena do jednotlivých rolí. Uživatele pak stačí pouze logicky zařadit do této struktury a přiřadit jim odpovídající role. Je však třeba dbát na to, že každý uživatel může mít přiřazen nejvýše takový počet rolí, jež je definován proměnnou MAX_ENABLED_ROLES, mezi přiřazené role se pak počítají i sub role. Proměnná MAX_ENABLED_ROLES je defaultně nastavena na hodnotu třicet, avšak může nabývat hodnot od nuly do sto čtyřiceti osmi.[4]

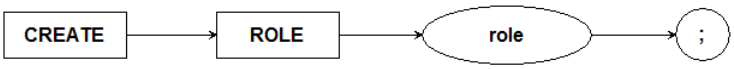
Příklad jednoduché hierarchické struktury rolí je znázorněn obrázkem 5.5. Jestliže následně v hierarchické struktuře rolí chceme změnit oprávnění nějaké skupiny uživatelů, stačí provést změny pouze v rámci rolí samotných a není tedy nutné měnit každému uživateli jeho oprávnění manuálně.

Například, jestliže v ukázkové hierarchii chceme uživatelům ze skupiny B přiřadit nové oprávnění, kupříkladu D, můžeme jednoduše toto oprávnění přiřadit k roli B, čímž oprávnění D získají nejen uživatelé ze skupiny B, ale též uživatelé ze skupiny A, kteří v ukázkové hierarchii plní roli jakýchsi kontrolních pracovníků.

V podnikových databázích pak bývají tyto hierarchické struktury často velmi rozsáhlé a je zde vyvíjena snaha o vytvoření takové hierarchické struktury, která by co nejvíce odpovídala skutečné hierarchii podniku. Díky využití rolí je tedy možné ušetřit značné množství zdrojového kódu a tím i času a finančních prostředků, zvláště v případech databází větších



Obrázek 5.5: Hierarchická struktura rolí
Zdroj: vlastní zpracování podle Oracle Database Help Center.[6]

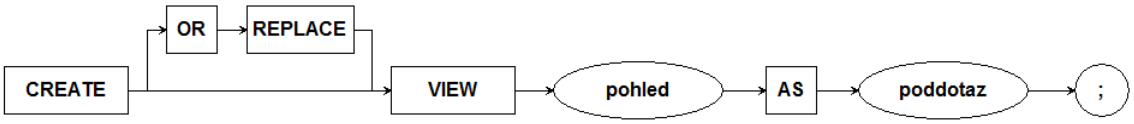


Obrázek 5.6: Struktura SQL příkazu CREATE ROLE
Zdroj: Vlastní zpracování.

společností, u nichž může vznikat potřeba změny oprávnění poměrně často. Další výhodou rolí je, že takto vytvořená hierarchie se snáze používá a také se v ní lépe provádí změny.

Pro vytváření rolí se využívá příkazu CREATE ROLE, základní struktura tohoto SQL příkazu je graficky znázorněna na obrázku 5.6. Jednotlivé role pak můžeme uživatelům přiřazovat pomocí příkazu GRANT nebo odebírat příkazem REVOKE, stejně jako je tomu u samostatných oprávnění, struktura příkazů GRANT a REVOKE pro role je stejná jako v případě systémových oprávnění viz 5.2.2, jejichž strukturu pak znázorňuje obrázek 5.3. Kromě námi vytvořených rolí existují v databázích Oracle také předpřipravené role, jež usnadňují některé úkony nebo například zajišťují kompatibilitu s dřívějšími verzemi softwaru Oracle.[4]

Jestliže chceme některou z rolí o něco více zabezpečit, je možné k ní přiřadit heslo, stejně jako je tomu u uživatelských účtů, jestliže využijeme tuto možnost, pak je nutné, aby uživatel využívající takovouto roli vždy zadal správné heslo. Heslo lze k roli přiřadit za pomoci příkazu ALTER ROLE v klauzuli IDENTIFIED stejně jako je tomu u uživatelských účtů. Toto heslo lze pak odebrat použitím klauzule NOT IDENTIFIED. Z toho důvodu, že větší



Obrázek 5.7: Struktura SQL příkazu CREATE VIEW
Zdroj: Vlastní zpracování.

množství hesel by značně snížilo uživatelské pohodlí, je nutné pečlivě volit, kdy tuto možnost využít. V praxi se většinou využívá pouze zřídka a jen u rolí, jež poskytují oprávnění k extrémně důležitým datům.[4]

5.3 Pohledy

Jedno ze základních Coddových pravidel pro relační model říká, že každý Systém řízení báze dat, musí umožňovat vytváření pohledů a práci s nimi. Proč je však toto pravidlo tak významné, je možné pochopit pouze, víme-li, co jsou to pohledy a k čemu slouží.

Pohledy neboli uložené dotazy, jak bývají občas nazývány, jsou virtuálními tabulkami a jako takové nejsou fyzicky ukládány na paměťová média, jako je tomu u běžných relačních tabulek. Naopak, pohledy jsou uchovávány pouze ve formě SQL dotazu, jenž definuje konkrétní pohled, z tohoto dotazu je následně vytvořena výsledná tabulka pokaždé znovu, je-li k pohledu přistupováno, díky tomu pohledy zabírají v paměti minimum místa.[7]

Pohledy jsou vytvářeny prostřednictvím příkazu CREATE VIEW, základní struktura tohoto SQL příkazu je graficky znázorněna na obrázku 5.7. Do klauzule poddotaz je pak možné umístit libovolný příkaz typu SELECT, díky čemuž mohou být pohledy založeny na datech z jedné nebo více tabulek a dokonce lze vytvářet i pohledy založené na datech z jiných pohledů. Pohledy mohou také obsahovat nové neznámé hodnoty, jež jsou dopočítány ze známých dat v databázi. Tabulkám, ze kterých jsou pohledy vytvořeny, se pak říká základní tabulky. Pohledy tak slouží jako určitá maska, jenž z celkové množiny informací, které je možné v dané databázi nalézt, odkrývá pouze takové, jenž jsou v dané situaci zapotřebí.[1]

Právě díky své velké variabilitě mají pohledy široké spektrum využití a můžeme je nalézt takřka v každé databázi. Hlavní využití pohledů spočívá především v modifikaci prezentovaných dat, čehož se hojně využívá v rámci tvorby bezpečnostních opatření. Při tvorbě bezpečnostních opatření se pak pohledy kombinují nejčastěji právě s mechanismem rolí, který byl popsán v předchozí části. Kombinace pohledů a rolí umožňuje naprosto přesně specifikovat kompetence jednotlivých uživatelů, čímž chrání data před přístupem neautorizovaných osob,

což snižuje riziko zneužití, zcizení či poškození významných dat. Tohoto efektu je dosaženo vytvořením pohledů, jež obsahují pouze data nezbytná k provádění konkrétních úkonů, ty jsou následně přiřazeny k rolím namísto základních tabulek. Každému z uživatelů je pak možné přidělit práva přesně k datům, jež odpovídají jeho informačním potřebám v rámci jeho pracovní činnosti. Díky takovýmto opatřením lze zabránit zneužití dat mimo rozsah kompetence konkrétního uživatele. Množství pohledů v databázi se může lišit v závislosti na potřebách jednotlivých podnikatelských subjektů.[8]

Další využití pohledů spočívá v podpoře uživatelů bez znalosti dotazovacích jazyků. Takovýto typ uživatelů dnes tvoří většina lidí, kteří přichází do styku s databázemi. Na míru vytvořené pohledy podle konkrétních specifikací mohou takovýmto uživatelům značně usnadnit práci s databází. Pro usnadnění přístupu pak bývají pohledy velmi často zabudovány do konkrétních databázových aplikací, které k jejich prezentaci využívají uživatelsky přívětivější prostředí.[2]

Kromě výhod však mají pohledy i jednu podstatnou nevýhodu. Jestliže vytvoříme příliš složitou hierarchii navzájem provázaných pohledů, může každý přístup znamenat vytváření velkého počtu na sebe navazujících pohledů, což nevyhnutelně povede ke zpomalení celého systému. Nicméně i přes tento fakt jsou pohledy velmi často využívány.

V relačních databázích se můžeme setkat se dvěma typy pohledů, jedním z nich jsou klasické pohledy. Jedná se o základní typ pohledů využívaných v relačních databázích, nevyžadují alokaci prostoru na paměťovém médiu a jsou uloženy pouze v datovém slovníku jako SQL dotazy definující konkrétní pohledy. Tyto SQL dotazy jsou vyvolány pokaždé, je-li vyžadován daný pohled. Dalším typem jsou pak takzvané materializované pohledy. Stejně jako u klasických pohledů i materializované pohledy jsou uloženy v datovém slovníku jako SQL dotazy definující konkrétní pohledy. Na rozdíl od nich jsou však materializované pohledy schopny alokovat místo na paměťovém médiu, na nějž materializovaný pohled uloží výsledek základního dotazu z datového slovníku, díky čemuž se při následném vyvolání materializovaného pohledu nemusí provádět celý dotaz znovu, ale stačí provést přírůstkovou aktualizaci, což při velkém množství dat může citelně zrychlit práci s databází. Další výhodou materializovaných pohledů je možnost replikace kopie tabulky do jiné databáze se stejnou definicí sloupců a dat jako u základní tabulky. Replikované tabulky jsou však pouze pro čtení.[2]

6 Návrh konkrétního řešení bezpečnosti

V této části se budeme zabývat praktickým využitím teoretických východisek, jež byla popsána v předchozích kapitolách, konkrétně zabezpečením databáze za pomoci pohledů, oprávnění, rolí, uživatelských účtů a profilů. Jako ukázkové prostředí nám bude sloužit databáze imaginárního podnikatelského subjektu, jenž je popsaný v části 6.1. Modelová databáze i prvky jejího zabezpečení jsou v rámci rozsahu bakalářské práce zredukovány pouze na ukázky vystihující podstatu popsaných teoretických východisek, v reálném podniku by se jednalo o mnohonásobně rozsáhlejší projekt. Veškeré zdrojové kódy jsou optimalizovány pro použití v prostředí Oracle Database 11g.

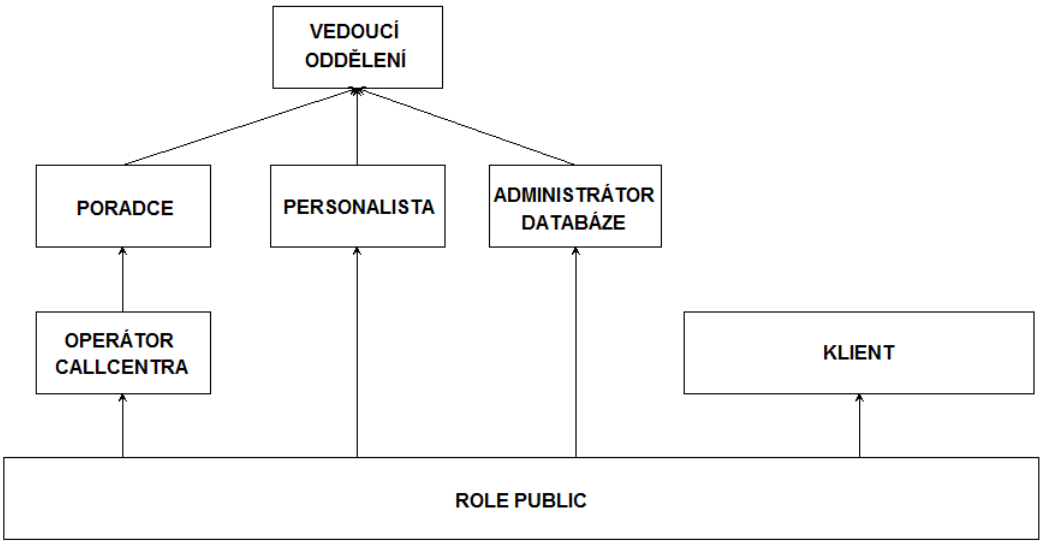
6.1 Popis modelové společnosti

Modelová společnost je malým až středně velkým podnikatelským subjektem, působícím na českém trhu od roku 2000. Sídlo společnosti a většina jejích poboček se nachází v Praze, kde má také svou hlavní klientskou základnu. Hlavní podnikatelskou činností společnosti je poskytování internetových a telekomunikačních služeb včetně prodeje přidružených zařízení. Cílem podniku je postupně rozšiřovat svou zákaznickou základnu díky kvalitě poskytovaných služeb a následně expandovat z Prahy a okolí i do dalších větších měst napříč Českou republikou.

V závislosti na typu podnikatelské činnosti je pro společnost důležité uchovávat v databázi informace o svých zákaznících a smlouvách, které se společností uzavřeli, dále pak o nabízených produktech, ale také informace o vlastních zaměstnancích, jednotlivých pobočkách a odděleních. Pro společnost je velmi důležité, aby informace obsažené v databázi byly na jednom místě a rychle dostupné i prostřednictvím internetu, avšak zároveň musí být zpřístupňovány uživatelům pouze na základě jejich autorizace podle bezpečnostního protokolu společnosti.

Uživateli firemní databáze jsou především zaměstnanci na různých pracovních pozicích, jako například operátoři call centra, poradci, personalisté, administrátoři databáze či vedoucí jednotlivých oddělení, dále také klienti či návštěvníci webových stránek. Podle potřeb těchto skupin jsou následně modelovány pohledy, role a také nastavovány uživatelské účty, příklady těchto bezpečnostních prvků jsou k nalezení v části 6.3.

Uživatelé podnikové databáze jsou zařazeni do hierarchické struktury rolí, jež vychází ze skutečného rozdělení podniku a jeho bezpečnostního protokolu. Jak by mohlo vypadat hierarchické rozdělení rolí v modelové společnosti je graficky znázorněno na obrázku 6.1.



Obrázek 6.1: Hierarchické rozdělení rolí
Zdroj: Vlastní zpracování.

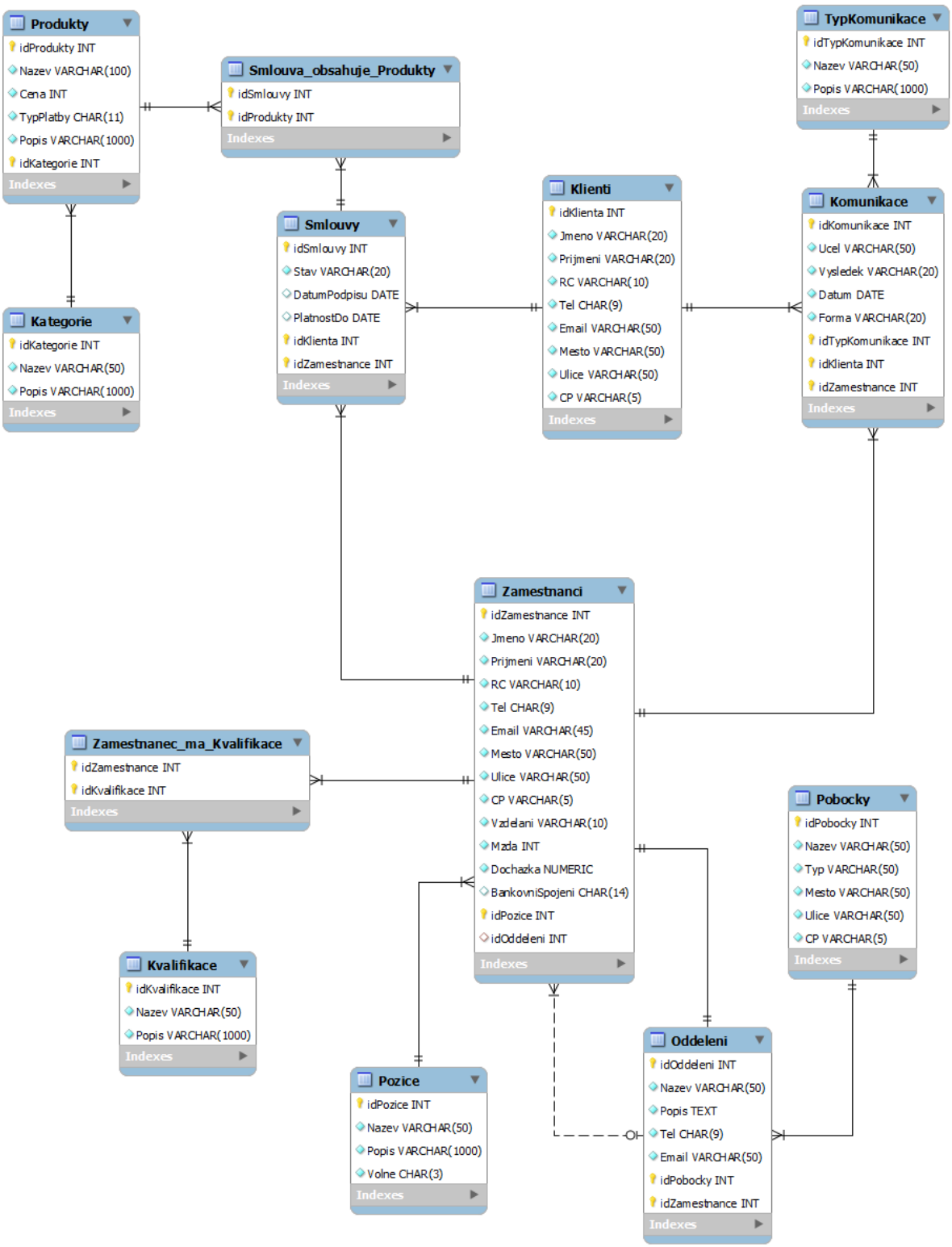
6.2 Návrh databáze

Ukázková databáze modelové společnosti je rozdělena na dvě části. První část je zaměřena na data týkající se hlavní podnikatelské činnosti společnosti a zahrnuje tabulky: Klienti, Smlouvy, Produkty, Komunikace, Kategorie, Typ komunikace a Smlouva_obsahuje_Produkty. Druhá část je pak zaměřena na informace o firmě samotné a jejích zaměstnancích, tato část zahrnuje tabulky: Zamestnanci, Pobocky, Oddeleni, Kvalifikace, Pozice a Zamestnanec_ma_Kvalifikace.

Schématické znázornění jednotlivých tabulek ukázkové databáze, jejich atributů a vztahů mezi nimi zachycuje obrázek 6.2, SQL skript pro vytvoření databáze je pak obsahem přílohy A.

6.3 Zabezpečení databáze

Ještě předtím, než je možné začít s vytvářením bezpečnostních prvků, je třeba nejprve zhodnotit rizika, jež by mohla databázi reálně ohrozit. Jelikož jsou v ukázkové databázi uchovávána data důvěrného charakteru o klientech a zaměstnancích společnosti, je třeba ji důkladně zabezpečit před narušením soukromí. Kromě osobních dat však databáze obsahuje také důležité informace o smlouvách či interní struktuře společnosti, ty by také měly podléhat utajení. Společnost navíc vyžaduje přístup k databázi prostřednictvím internetu, tím se však vystavuje riziku napadení z vnějšku. S ohledem na tyto faktory je třeba realizovat interní i externí



Obrázek 6.2: Schématické znázornění databáze modelované společnosti
Zdroj: Vlastní zpracování.

bezpečnostní prvky. Aby bylo možné zabránit zneužití již zmíněných dat, je třeba zajistit, aby se k těmto datům nedostaly neautorizované osoby, jež by mohly narušit bezpečnost databáze a zcizit či znehodnotit data v ní uložená. V rámci zabezpečení databáze je dosaženo bezpečnosti dat podléhajících utajení následujícím způsobem:

1. Nejprve je třeba definovat datové potřeby jednotlivých uživatelů při konkrétních činnostech, podle nich následně vytvořit pohledy, jež budou v kombinaci s oprávněními přesně vymezovat kompetence jednotlivých uživatelů. Díky tomu bude mít každý přístup pouze k datům, s nimiž je oprávněn pracovat.

2. Dále je nutné rozdělit uživatele na skupiny v závislosti na jejich pozici ve společnosti. Podle takto definovaných uživatelských skupin pak vytvořit hierarchickou strukturu rolí, s jejichž pomocí definujeme, k jakým činnostem jsou v rámci databáze jednotliví uživatelé oprávněni. Každého nového uživatele je třeba následně zařadit do některé z uživatelských skupin a přidělit mu odpovídající role, popřípadě, je-li to nezbytně nutné, i další doplňující oprávnění.

3. Nakonec je nutné za pomoci profilů definovat bezpečnostní parametry uživatelských účtů, jež budou odpovídat důležitosti dat, k nimž by daný uživatel mohl mít přístup. Tímto krokem snížíme riziko získání kontroly nad cizím uživatelským účtem neautorizovanou osobou a následného zcizení či znehodnocení dat.

Databázi je samozřejmě třeba chránit i před dalšími hrozbami jako jsou například ztráta integrity či dostupnosti, které jsou popsány v části 5.1, to však již není předmětem této ukázky.

6.3.1 Pohledy

Jednotliví uživatelé jsou v rámci databáze oprávněni vykonávat odlišné úkony. Aby bylo zajištěno, že se při práci s databází dostane každý pouze k datům, k nimž je autorizován přistupovat, čímž lze snížit riziko úniku informací, je vhodné nejprve vytvořit pohledy. Pohledy mohou být různě složité v závislosti na datech, jež ke své práci uživatelé potřebují. V rámci ukázkové databáze bude vytvořeno několik pohledů od nejjednodušších až po složitější.

Nejjednodušší pohledy se omezují pouze na data v rámci jediné tabulky. Jako příklad poslouží následující pohled, jenž zaměstnancům společnosti poskytne základní kontaktní údaje o klientech. Konkrétně jméno, příjmení a telefonní číslo.

```
CREATE VIEW KlientiKontakt AS
SELECT Jmeno, Prijmeni, Tel AS Telefon
FROM Klienti;
```

Pohledy se však nemusí nutně omezovat pouze na jedinou tabulku, naopak velmi často využívají data hned z několika tabulek, jako je tomu u následujícího pohledu, ten zobrazí důležité informace o produktech, jenž společnost nabízí. Konkrétně název, cenu, o jaký typ platby se jedná a nakonec kategorii, do níž je produkt zařazen.

```
CREATE VIEW ProduktyKomplet AS
SELECT Produkty.Nazev AS Produkt, Cena, TypPlatby,
Produkty.Popis, Kategorie.Nazev AS Kategorie
FROM Produkty, Kategorie
WHERE Produkty.idKategorie = Kategorie.idKategorie;
```

V rámci pohledu je také možné využít složitější konstrukci příkazu SELECT, který pak může obsahovat větší množství podmínek nebo další poddotazy. Následující pohled zobrazí veškeré informace o zaměstnancích, kteří jsou podřízenými vedoucího oddělení, jenž tento pohled vyvolal.

```
CREATE VIEW ZamestnanciNaOddeleni AS
SELECT Zamestnanci.*, Nazev
FROM Zamestnanci, Pozice
WHERE Zamestnanci.idPozice = Pozice.idPozice
AND idOddeleni = (
SELECT idOddeleni
FROM Oddeleni
WHERE idZamestnance = (
SELECT idZamestnance
FROM Zamestnanci
WHERE Email = (SELECT USER FROM DUAL)));
```

V reálných databázích se však velmi často vyskytují mnohem složitější a komplexnější pohledy, jež spojují větší množství tabulek, obsahují mnoho poddotazů a navíc dopočítávají některé neznámé hodnoty. Poslední příklad demonstruje, jakou složitost mohou mít pohledy v reálných databázích. Následující pohled tedy zobrazí veškeré informace o smlouvě, jméno, příjmení a rodné číslo klienta, ke kterému je smlouva přiřazena, dále pak celkovou měsíční a jednorázovou cenu veškerých položek ve smlouvě a také, kdo ze zaměstnanců smlouvu vytvořil.

```
CREATE VIEW SmlouvyPrehled AS
SELECT Smlouvy.*, Klienti.Jmeno, Klienti.Prijmeni, Klienti.RC,
Zamestnanci.Prijmeni AS Zamestnanec,
NVL(MesicniPoplatky, 0) AS MesicniPoplatky,
NVL(JednorazovePoplatky, 0) AS JednorazovePoplatky
FROM Klienti, Zamestnanci, Smlouvy
LEFT JOIN (
SELECT ID1, NVL(Mesicne,0) AS MesicniPoplatky,
Celkem - NVL(Mesicne,0) AS JednorazovePoplatky
FROM ((
SELECT Smlouvy.idSmlouvy AS ID1, SUM(Cena) AS Celkem
FROM Produkty, Smlouvy, Smlouva_obsahuje_Produkty
WHERE Smlouvy.idSmlouvy = Smlouva_obsahuje_Produkty.idSmlouvy
AND Produkty.idProduktu = Smlouva_obsahuje_Produkty.idProduktu
GROUP BY Smlouvy.idSmlouvy)
LEFT JOIN (
SELECT Smlouvy.idSmlouvy AS ID2, SUM(Cena) AS Mesicne
FROM Produkty, Smlouvy, Smlouva_obsahuje_Produkty
WHERE Smlouvy.idSmlouvy = Smlouva_obsahuje_Produkty.idSmlouvy
AND Produkty.idProduktu = Smlouva_obsahuje_Produkty.idProduktu
AND typplatby = 'mesicni'
GROUP BY Smlouvy.idSmlouvy)
ON ID1 = ID2 )) ON Smlouvy.idSmlouvy = ID1
WHERE Smlouvy.idKlienta = Klienti.idKlienta
AND Smlouvy.idZamestnance = Zamestnanci.idZamestnance;
```

6.3.2 Role

Po vytvoření pohledů je dalším nezbytným krokem tvorba jednotlivých rolí, bez nichž by prakticky nebylo možné zajistit efektivní zabezpečení databáze. Tím jaká oprávnění uživatelům přidělíme totiž definujeme úkony, jež mohou konkrétní uživatelé s jednotlivými objekty v databázi provádět. Přidělovaná oprávnění by pak měla odpovídat jejich zařazením v rámci společnosti. Jestliže má mít databáze větší množství uživatelů, je vhodné jednotlivá oprávnění seskupit do rolí, čímž se značně sníží množství použitého kódu a navíc je zredukována i možnost přidělení špatných oprávnění. V této části budou pro ukázku definovány tři uživatelské skupiny a vytvořeny pro ně specifické role.

Operátoři call centra

Jde o nové zaměstnance společnosti pracující na oddělení styku se zákazníky, jejich hlavní pracovní náplní je nabízení produktů klientům z databáze a uzavírání nových smluv, touto

prací se seznamují s produkty, jež společnost nabízí a také se učí jednat s klienty. Operátoři mají přístup k informacím o produktech a kontaktních údajích o klientech, mohou také vkládat do databáze nové smlouvy a jejich obsah. V závislosti na jejich pracovní náplni je pro operátory vytvořena následující role.

```
CREATE ROLE Callcentrum;
GRANT SELECT ON ProduktyKomplet TO Callcentrum;
GRANT SELECT ON KlientiKontakt TO Callcentrum;
GRANT INSERT ON Smlouvy TO Callcentrum;
GRANT INSERT ON Smlouva_obsahuje_Produkty TO Callcentrum;
GRANT CREATE SESSION TO Callcentrum;
```

Poradci

Zaměstnanci společnosti pracující na oddělení styku se zákazníky, kteří již jsou u společnosti delší dobu a mají tak více zkušeností než operátoři call centra, také prošli různými školeními, které firma organizuje. Mezi jejich hlavní pracovní náplň patří získávání nových klientů a uzavírání smluv, komunikace se stávajícími klienty a poskytování odborného poradenství. Poradci mají stejná práva jako operátoři call centra, navíc však mají přístup ke kompletním informacím o klientech, jejich smlouvách a také k záznamům o komunikaci s klienty. V závislosti na jejich pracovní náplni je pro poradce vytvořena následující role.

```
CREATE ROLE Poradce;
GRANT Callcentrum TO Poradce;
GRANT SELECT ON SmlouvyPrehled TO Poradce;
GRANT SELECT, INSERT, UPDATE ON Klienti TO Poradce;
GRANT SELECT, INSERT, UPDATE ON Komunikace TO Poradce;
GRANT UPDATE ON Smlouvy TO Poradce;
GRANT UPDATE ON Smlouva_obsahuje_Produkty TO Poradce;
GRANT CREATE SESSION TO Poradce;
```

Vedoucí oddělení

Zaměstnanci společnosti na manažerských pozicích, kteří zodpovídají za chod jim svěřeného oddělení. V rámci této činnosti mají přístup k veškerým informacím o svých podřízených. Vedoucí oddělení však navíc mohou provádět i veškeré úkony jako jejich podřízení. V závislosti na jejich pracovní náplni je pro vedoucí oddělení vytvořena následující role, jež doplňuje manažerské oprávnění k ostatním rolím, jimiž vedoucí oddělení disponuje.

```
CREATE ROLE Vedouci;
GRANT SELECT ON ZamestnanciNaOddeleni TO Vedouci;
GRANT INSERT, UPDATE, DELETE ON Zamestnanci TO Vedouci;
GRANT CREATE SESSION TO Vedouci;
```

V důsledku zavedení kombinace rolí a pohledů, je možné přesně definovat, kdo je autorizován k práci s jakými objekty. Pokud by například některý z poradců chtěl vyvolat pohled SmlouvyPrehled systém by mu potřebná data zpřístupnil ve formě relační tabulky. Avšak jestliže by se naopak snažil vyvolat pohled ZamestnanciNaOddeleni, jenž je specifický pro vedoucí oddělení, systém by takovýto požadavek neprovedl a ohlásil by následující chybu:

```
ORA-00942: table or view does not exist
```

Tímto způsobem tedy lze zajistit interní bezpečnost databáze, neboli, že se uživatelé nedostanou v rámci databáze za hranice svých kompetencí. V reálném podniku by každá skupina uživatelů měla obdobné role, jenž by definovaly jejich práva. Většina uživatelů firemních databází pak disponuje pouze objektovými oprávněními, výjimku tvoří většinou zaměstnanci starající se o rozvoj databáze, ti zpravidla disponují i právy systémovými.

6.3.3 Profily

Posledním krokem při tvorbě efektivního zabezpečení je tvorba externí bezpečnosti, v našem případě nastavení jednotlivých profilů. Profily najdou uplatnění zejména u veřejně dostupných databází, v nichž jsou uchovávána důležitá data, která je nutné utajit před neautorizovanými osobami. Především v takovýchto případech je třeba zajistit jistou míru zabezpečení uživatelských účtů, aby nedošlo k jejich zneužití za účelem získání klíčových dat. Zabezpečení uživatelského účtu je možné zajistit právě přiřazením profilu, jenž definuje potřebné bezpečnostní prvky. V této části bude vytvořen jako příklad zaměstnanecký profil.

Jelikož zaměstnanci modelové společnosti mají přístup k nejrůznějším informacím důvěrného charakteru, například citlivá data o klientech a jejich smlouvách, je vhodné zajistit určitou míru zabezpečení jejich uživatelských účtů. V procesu vytváření bezpečnostních prvků je však třeba brát v potaz nejen bezpečnost, ale také uživatelské pohodlí při práci s databází a následně nalézt mezi těmito aspekty kompromis, proto zaměstnanecký profil nastavíme následujícím způsobem.

```
CREATE PROFILE zamestnanec LIMIT
SESSIONS_PER_USER          1
FAILED_LOGIN_ATTEMPTS      3
PASSWORD_LOCK_TIME         1
CONNECT_TIME               480
IDLE_TIME                  30
PASSWORD_LIFE_TIME         90
PASSWORD_GRACE_TIME        10
PASSWORD_REUSE_MAX         4
PASSWORD_REUSE_TIME        365
PASSWORD_VERIFY_FUNCTION   Custom_Password_Function;
```

Díky takto nastavenému profilu může zaměstnanec zůstat přihlášen osm hodin, aby mohl nerušeně pracovat po celou svou pracovní dobu, nicméně po třiceti minutách nečinnosti bude od systému automaticky odhlášen. V rámci bezpečnosti může mít zaměstnanec jen jednu aktivní relaci a pro přihlášení má pouze tři pokusy na zadání správného hesla, než bude účet uzamčen. Uzamčený účet se pak automaticky reaktivuje po uplynutí jednoho dne nebo jej může opět aktivovat administrátor databáze. Od zaměstnance je dále vyžadováno, aby nejpozději každých devadesát dní měnil své heslo. Nové heslo pak musí být od předchozího odlišné. Stejné heslo lze použít až po čtyřech změnách a uplynutí minimálně jednoho roku. Každé heslo navíc musí být minimálně osm znaků dlouhé a musí obsahovat alespoň jedno písmeno a číslo, toho je dosaženo za pomoci kontrolní funkce Custom_Password_Function.

Aby zaměstnanecký profil správně fungoval je tedy nejprve potřeba definovat funkci pro kontrolu hesel, ta musí být typu BOOLEAN a musí mít tři vstupní parametry typu VARCHAR2, to je dáno standardem Oracle.[6]

```
CREATE FUNCTION Custom_Password_Function(
username VARCHAR2,
password VARCHAR2,
old_password VARCHAR2)

RETURN BOOLEAN AS
--definice pomocných proměnných
cislo BOOLEAN;
znak BOOLEAN;
digitarray VARCHAR2(10);
chararray VARCHAR2(52);

BEGIN
--inicializace pomocných proměnných
  digitarray := '0123456789';
  chararray := 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

```
  cislo := FALSE;
  znak := FALSE;

  --porovnání délky hesla
  IF LENGTH(password) < 8 THEN
    raise_application_error(-20000,
        'Heslo je kratší než 8 znaků');
  END IF;

  --hledání čísla v hesle
  FOR i IN 1..LENGTH(digitarray) LOOP
    FOR j IN 1..LENGTH(password) LOOP
      IF substr(password,j,1) = substr(digitarray,i,1) THEN
        cislo := TRUE;
        GOTO hledejznak;
      END IF;
    END LOOP;
  END LOOP;

  IF cislo = FALSE THEN
    raise_application_error(-20001,
        'Heslo neobsahuje žádné čislo');
  END IF;

  --hledání písmene v hesle
  <<hledejznak>>
  FOR i IN 1..LENGTH(chararray) LOOP
    FOR j IN 1..LENGTH(password) LOOP
      IF substr(password,j,1) = substr(chararray,i,1) THEN
        znak := TRUE;
        GOTO konec;
      END IF;
    END LOOP;
  END LOOP;

  IF znak = FALSE THEN
    raise_application_error(-20002,
        'Heslo neobsahuje žádné písmeno');
  END IF;

  <<konec>>
  RETURN TRUE;
  END Custom_Password_Function;
/
```

V reálných databázích mohou být takovéto funkce i mnohem rozsáhlejší, aby odpovídaly požadavkům konkrétního zadavatele. Vždy je však třeba přizpůsobit vynucovanou složitost hesel tomu, s jak významnými daty přichází uživatel do styku.

Pro ilustraci funkčnosti vytvořeného profilu je možné uvést několik příkladů, jak bude databázový systém reagovat na potenciálně nebezpečné chování uživatelů. V případě, že by uživatel zadal neplatné heslo, ve snaze o přihlášení k systému, bude mu přístup odepřen. Nemuselo by se se totiž jednat o autorizovaného uživatele, nýbrž o osobu snažící se prolomit bezpečnostní opatření databáze. Toto chování systému by bylo odůvodněno chybovou hláškou:

```
ORA-01017: invalid username/password; logon denied
```

Jestliže by se uživatel třikrát v řadě neúspěšně pokusil o přihlášení, jeho uživatelský účet by byl zablokován na dobu nastavenou parametrem PASSWORD_LOCK_TIME. Reaktivaci účtu může uspíšit pouze administrátor klauzulí ACCOUNT UNLOCK v příkazu ALTER USER. Při dalších pokusech o přihlášení k zablokovanému účtu by pak systém hlásil chybu:

```
ORA-28000: the account is locked
```

Pokud by chtěl uživatel po úspěšném přihlášení změnit své heslo, je nutné aby dodržel veškerá pravidla nastavená přiděleným profilem. Jestliže by například chtěl použít stejné heslo dříve, než uplyne doba nastavená parametrem PASSWORD_REUSE_TIME nebo před provedením minimálního počtu změn daných parametrem PASSWORD_REUSE_MAX, systém by mu v takovémto počínání zabránil a nahlásil by chybu:

```
ORA-28007: the password cannot be reused
```

Pokud by zvolil heslo nové, které by však neodpovídalo parametrům funkce pro verifikaci hesel, takováto změna by také neproběhla a systém by nahlásil chybu selhání verifikační funkce a jednu z dodatečných chyb, jež jsou definovány přímo ve funkci:

```
ORA-28003: password verification failed
ORA-20000: Heslo je kratší než 8 znaků
ORA-20001: Heslo neobsahuje žádné číslo
ORA-20002: Heslo neobsahuje žádné písmeno
```

Jestliže uživatel zůstane v nečinnosti déle než je nastaveno parametrem IDLE_TIME nebo je již připojen déle než udává limit CONNECT_TIME, systém mu nedovolí pokračovat v další

práci dříve, než po opětovném přihlášení. Tímto je možné zabránit získání kontroly nad uživatelským účtem, jenž byl ponechán přihlášen bez dozoru. Systém by v takovémto případě nahlásil jednu z chyb:

```
ORA-02396: exceeded maximum idle time
ORA-02399: exceeded maximum connect time
```

Jako poslední vzorovou situaci lze uvést vypršení doby platnosti hesla. Tato situace nastane ve chvíli, kdy dojde k překročení doby nastavené parametrem PASSWORD_LIFE_TIME. Systém zareaguje nastavením stavu hesla jako „expired", takže při pokusu o přihlášení bude muset uživatel změnit své heslo. Jestliže ke změně nedojde po dobu nastavenou parametrem PASSWORD_GRACE_TIME, platnost hesla definitivně vyprší a již nebude možné se přihlásit. Průběh takovéto situace pak vypadá následovně:

```
ORA-28001: the password has expired

Changing password for "uživatel"
New password:
Retype new password:

Pasword changed
Connected.
```

6.3.4 Životní cyklus uživatelského účtu

V této části bude na vzorovém uživateli, panu Novákovi, demonstrováno, jakým způsobem by mohl vypadat životní cyklus konkrétního uživatelského účtu. Veškeré zde použité objekty jsou popsány v předchozích částech této kapitoly.

Pan Novák úspěšně prošel výběrovým řízením a díky jeho znalostem a předchozím zkušenostem mu byla v modelové společnosti nabídnuta pozice poradce. Poté, co byly informace o jeho osobě vloženy do databáze, byl panu Novákovi vytvořen nový uživatelský účet a přidělena standardní oprávnění poradce:

```
CREATE USER novak@company.cz
IDENTIFIED BY X45b2DFG PROFILE zamestnanec;
GRANT Poradce TO novak@company.cz;
```

Po několika letech, kdy se již pan Novák osvědčil jako poradce, se uvolnilo místo vedoucího oddělení pro styk se zákazníky. Na základě rozhodnutí vedení společnosti byl po dobu trvání

výběrového řízení pověřen právě pan Novák. V rámci této dočasné personální změny mu byla přidělena další oprávnění, aby mohl vykonávat příslušné operace:

```
GRANT Vedouci TO novak@company.cz;
```

Po zakončení výběrového řízení, ve kterém byl vybrán nový vedoucí oddělení pro styk se zákazníky, byl pan Novák ve funkci vedoucího oddělení nahrazen a zařazen zpět na pozici poradce. Současně s tím mu byla odebrána oprávnění vedoucího oddělení:

```
REVOKE Vedouci FROM novak@company.cz;
```

Následující rok pan Novák dostal výhodnou nabídku práce, kterou se rozhodl přijmout. Na základě jeho výpovědi byla po ukončení pracovního poměru odebrána veškerá data o jeho osobě z databáze a také byl zrušen jeho uživatelský účet:

```
DROP USER novak@company.cz;
```

Je možné si povšimnout, že ve chvíli, kdy jsou již jednotlivé bezpečnostní prvky vytvořeny, práce s uživatelskými je účty poměrně jednoduchá. Velkou úsporu kódu pak přináší zejména tvorba hierarchické struktury rolí. Při nízkém počtu uživatelů je možné se mylně domnívat, že tato úspora nebude nijak významná, avšak opak je pravdou. Například, již při dvaceti uživatelích na pozici poradce by bylo ušetřeno použitím rolí sto třicet dva řádek kódu jen na přidělení patřičných oprávnění. Toto číslo je pak v reálných databázích s velkým počtem uživatelů mnohem vyšší.

6.4 Srovnání použitých nástrojů

V rámci navrhovaného řešení byly využity tři rozdílně nástroje, s nimiž se lze běžně v praxi setkat při tvorbě zabezpečení relačně databázových evidencí. Nyní je třeba tyto nástroje porovnat, zhodnotit jejich přínosy a uvést, kdy je vhodné je využít. Zde však vyvstává problém. Jakým způsobem lze porovnat techniky řešící různé části rozsáhlé problematiky, jejichž složitost je do značné míry závislá na konkrétních parametrech zadání, a které jsou navíc vytvořeny v neprocedurálním jazyce SQL? Jelikož použité nástroje neřeší stejný problém a jejich složitost je velmi proměnlivá, díky čemuž pro ně nelze vytvořit jakýsi etalon složitosti, srovnání mohutnosti kódu pomocí metody LOC by bylo irelevantní. Ze stejných důvodů je nutné zavrhnout i další metody, jako například porovnání časové složitosti, zde navíc vzniká problém závislosti práv a pohledů, které by pro vlastní měření nebylo možné jednoduše separovat. Není možné využít ani metodu porovnání cyklomatické složitosti, jelikož SQL je

neprocedurálním jazykem, který až na výjimky v podobě funkcí v kódu neobsahuje smyčky ani rozhodovací příkazy, a tak by nebylo možné tímto způsobem objektivně hodnotit jejich složitost. Z uvedených důvodů je patrné že využití kvantitativních metod srovnání není v tomto případě příliš vhodné, proto budou jednotlivé nástroje porovnávány kvalitativně. Konkrétně podle náročnosti na znalosti potřebné k jejich realizaci, části zabezpečení, v níž jsou využívány a podle jejich výsledného efektu. Na závěr pak bude zobecněno, kdy je třeba v podnikových databázích využívat jednotlivé techniky.

Jak může být patrné z ukázkových kódů, nejnáročnější jsou na vstupní znalosti právě profily. Samotné nastavení profilů sice není nikterak náročné a je nutné znát pouze vlastnosti jednotlivých limitů, složitost profilů se však projeví ve chvíli, kdy je třeba vytvořit verifikační funkci, s jejíž pomocí lze zabránit uživatelům v používání triviálních hesel. Zde je již třeba mít pokročilé znalosti SQL a především pak ovládat tvorbu funkcí. Druhé v pořadí, co se týče potřebných znalostí, jsou pohledy. Každý pohled je vždy založen na příkazu SELECT, a proto je k jejich realizaci potřeba dobře rozumět logice dotazování v relačních databázích. U složitějších pohledů je pak důležité znát i pokročilé mechanismy využívané při vyhledávání, jako jsou různé typy spojování tabulek či agregační funkce. Poslední a tedy nejméně náročné jsou oprávnění, respektive role. K jejich realizaci je vesměs zapotřebí znát pouze význam jednotlivých oprávnění a s těmito znalostmi již lze konstruovat libovolné role.

Ač řeší různé části zabezpečení, pro všechny popsané nástroje platí to samé, chceme-li je využívat korektně, je nejprve nutné správně analyzovat čeho chceme jejich použitím dosáhnout. Je tedy třeba zmínit, kde se jednotlivé nástroje využívají a jaký je jejich výsledný efekt.

Princip oprávnění, respektive rolí, je nejrozšířenějším nástrojem pro tvorbu interního zabezpečení, to je způsobeno následujícím faktem. Bez přiřazení oprávnění nelze v databázi provádět žádné operace, a tak je jejich využití naprosto nezbytné. Hlavní úlohou rolí je vymezit kompetence jednotlivým autorizovaným uživatelům databáze. Díky přidělování práv je tedy možné zabránit uživatelům v přístupu k datům, s nimiž nejsou oprávněni pracovat.

Pohledy jsou stejně jako role využívány při tvorbě interního zabezpečení. V databázích se využívají velmi hojně, především proto, že jejich využití netkví pouze v tvorbě zabezpečení. Pohledy umožňují modifikovat prezentovaná data, což v kombinaci s rolemi zajišťuje mnohem přesnější cílení práv na konkrétní uživatelské skupiny. Takto lze do značné míry zvýšit efektivitu oprávnění a tedy i interní bezpečnosti. Jak může být patrné, výsledný efekt využití pohledů v rámci zabezpečení spočívá spíše v podpoře mechanismu oprávnění. Samy o sobě totiž pohledy neposkytují žádnou úroveň zabezpečení, jelikož nedokáží zajistit, že uživatelé budou využívat právě pohledy namísto základních tabulek, k tomu je již třeba využít i oprávnění.

Profily jsou na rozdíl od již zmíněných nástrojů využívány k zajištění externí bezpečnosti. V rámci zabezpečení databáze je pak primárním úkolem profilů zajistit snížení bezpečnostních rizik spojených se zneužitím uživatelského účtu neautorizovanou osobou. Takovéhoto efektu je dosaženo posílením bezpečnostních prvků uživatelských účtů, s jejichž pomocí je pak možné prosazovat dodržování námi definovaných zásad bezpečnosti.

Který z představených nástrojů je vhodné nasadit při realizaci zabezpečení databáze pro docílení optimální ochrany? To záleží jednak na specifikaci konkrétního podniku, ale především pak na parametrech databáze, pro níž je zabezpečení vytvářeno. Při rozhodování se však obecně lze řídit následujícími poznatky.

- Oprávnění tvoří naprostý základ zabezpečení a musí být přítomna v každé databázi. Bez patřičných oprávnění by se totiž uživatelé nemohli k databázi ani přihlásit.
- Jestliže má mít databáze více než jen pár uživatelů, je vhodné sdružit jednotlivá oprávnění do hierarchické struktury rolí, čímž ušetříme značné množství zdrojového kódu a tedy i času a peněz.
- Pohledy je třeba využít zejména tam, kde jsou kladeny specifické nároky na kompetence uživatelů. V takovýchto případech již nestačí přidělovat práva k základním tabulkám, ale je nutné přesně definovat, k jakým datům má mít uživatel v rámci tabulek přístup.
- Profily by měly být využity především v případech, kdy je třeba řešit externí zabezpečení. Tedy je-li k databázi možné přistupovat prostřednictvím internetu a jsou-li v ní uchovávána důležitá data, jež musí být utajena, jako například citlivá či strategický významná data.

Z těchto poznatků je patrné, že menší podniky, k jejichž databázím má přístup pouze omezené množství zaměstnanců, si v podstatě mohou vystačit i s použitím samotných rolí. Jak se však společnost rozrůstá, postupně vzniká potřeba více členit jednotlivá oprávnění, což nutně znamená i použití pohledů. A nakonec, jedná-li se o podnik, jenž umožňuje přístup k databázi široké veřejnosti prostřednictvím internetu, měla by být zvážena možnost použití profilů. Rozhodnutí pak zpravidla závisí na důležitosti dat, jež jsou v databázi uchovávána a také na tom, zda jejich ztráta poškodí společnost nebo jen konkrétního uživatele.

7 Závěr

Bakalářská práce byla zaměřena na problematiku zabezpečení databázově evidovaných dat v podnikatelských subjektech. Hlavním cílem bakalářské práce bylo objasnit teoretické principy související s řešenou problematikou a vymezit její relevantnost. Dále pak navrhnout řešení této problematiky v souladu s identifikovanými požadavky a se zřetelem na reálné možnosti podnikatelských subjektů. K dosažení vytyčených cílů bakalářské práce byla použita metodika založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Stěžejní pro vypracování této závěrečné práce byly především nástroje a techniky využívané v rámci relačně databázové technologie v kontextu s problematikou zabezpečení.

Na počátku vytyčené cíle bakalářské práce byly splněny, přičemž ze studia a analýzy vybraných informačních zdrojů a následného návrhu a zpracování vlastního řešení zvolené problematiky vyplynuly následující důležité poznatky.

Síla informací je v dnešní době ve společnosti obecně velmi ceněna. Zejména pak proto, že v tržně ekonomické společnosti jsou informace pro podnikatelské subjekty nezbytným zdrojem konkurenceschopnosti. A právě kvůli síle informací je třeba, aby data, jež jsou jejich nositelem, byla efektivně chráněna před možným zneužitím či poškozením. Pokud by snad došlo k narušení bezpečnosti, mohly by neautorizované osoby získat přístup k citlivým či strategicky významným datům, jež společnost eviduje ve svých databázích. Jestliže by nastala podobná situace a důležitá data by byla zcizena či poškozena, pro podnikatelský subjekt by to mohlo znamenat ztrátu důvěry svých klientů, konkurenceschopnosti nebo i právní kroky proti společnosti. Aby bylo možné zabránit podobným situacím, je nutné navrhovat co možná nejbezpečnější databázové systémy, zejména jedná-li se o databáze podniků evidujících data, jež by měla podléhat utajení.

Předtím, než je vůbec možné začít vytvářet jednotlivé prvky zabezpečení relačně databázové evidence, je nejprve nutné vytvořit robustní databázi, která dokáže odolat nejrůznějším kritickým scénářům a při tom zachovat relevanci dat. Takováto databáze se poté může stát stabilní platformou pro konkrétní řešení bezpečnosti. K zajištění stability navrhovaných evidencí se v rámci relačně databázové technologie využívají především prvky jako jsou transakční zpracování, integritní omezení a v neposlední řadě též vlastnosti relačních tabulek, jež tvoří základ relačně datového modelu.

Při návrhu bezpečnostních opatření pro konkrétní databázi je vždy nejprve nutné si odpovědět na následující otázky. Jak důležitá data se snažíme chránit? Kdo bude autorizován pro

práci s databází, kterou se snažíme zabezpečit? Jaké budou kompetence jednotlivých uživatelů? Dále je třeba si uvědomit, jaká rizika mohou databázi reálně hrozit a odkud by mohlo dojít k narušení bezpečnosti. Zmíněné faktory ovlivňují, jaké nástroje při tvorbě zabezpečení použijeme, ale také směr, jímž se bude následné vývoj konkrétního řešení bezpečnosti ubírat. Jestliže jsou zmíněné faktory správně zanalyzovány, je možné nastavit optimální míru zabezpečení. Díky správnému nastavení bezpečnostních prvků lze databázi efektivně chránit a zároveň zabránit snížení pracovní efektivity uživatelů. Zabezpečení se dělí na interní a externí, přičemž interní bezpečnost se věnuje otázkám řešícím problematiku vymezení kompetencí autorizovaných uživatelů databáze. Naopak externí bezpečnost řeší, jakým způsobem zabránit neautorizovaným osobám v přístupu k databázi. Nejrozšířenějším nástrojem pro tvorbu interního zabezpečení jsou oprávnění, respektive role, jež jsou často kombinovány s pohledy za účelem zvýšení efektivity interní bezpečnosti. Tato kombinace umožňuje přesně vymezovat kompetence jednotlivých uživatelů. Při tvorbě externího zabezpečení se pak velmi často využívají profily, které rozšiřují možnosti zabezpečení uživatelských účtů.

Relačně databázová technologie, jejíž základ vytvořil E. F. Codd již v roce 1970, je i dnes po více než čtyřiceti letech stále rozvíjena a přináší nové poznatky v problematice zpracování hromadných dat a především pak nové metody usnadňující tvorbu databázového zabezpečení. Původní technologie se postupně rozdělila na tři vývojové směry, kterými jsou relační, objektově-relační a čistě objektové databáze. Každý z těchto modelů má své výhody i nevýhody. Avšak i přes fakt, že objektově orientované technologie jsou dnes velice populární, zůstává relačně databázová technologie stále nejrozšířenější alternativou pro podnikové systémy, v nichž je třeba evidovat velké množství dat. Do budoucna se pravděpodobně budou stále více rozvíjet objektově orientované technologie, potrvá však ještě dlouho, než relační databáze zcela vymizí.

Literatura

[1] BRYLA, Bob a Kevin LONEY. Mistrovství v Oracle Database 11g. Computer Press 2010. ISBN: 978-80-251-2189-4

[2] CONNOLY, Tomas. Mistrovství Databáze - Profesionální průvodce tvorbou efektivních databází. Computer Press, 2009. ISBN: 978-80-251-2328-7

[3] HERNANDEZ, Michael J. Návrh databází. GRADA, 2006. ISBN: 80-247-0900-7

[4] LONEY, Kevin. Oracle database - Kompletní průvodce. Computer Press 2010. ISBN: 978-80-251-2489-5

[5] OPPEL, Andy. Databases DeMYSTiFieD, 2nd Edition. McGraw-Hill Companies, 2010. ISBN: 978-0071747998

[6] Oracle Database Help Center. Database documentation. [online]. 21.11.2014 [cit. 2015-01-12]. Dostupné z: https://docs.oracle.com/en/database/

[7] POKORNÝ, Jaroslav a Michal VALENTA. Databázové systémy. Česká technika - vydavatelství ČVUT, 2013. ISBN: 978-80-01-05212-9

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

[9] VOSTROVSKÝ, Václav. Vytváření databází v Oracle. Česká zemědělská univerzita v Praze, 2009. ISBN: 978-80-213-1191-6

Seznam obrázků

Seznam tabulek

A DDL skript modelové databáze

```
CREATE TABLE Kategorie (
idKategorie NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (50) NOT NULL,
Popis VARCHAR2 (1000) NOT NULL);

CREATE TABLE Klienti (
idKlienta NUMBER NOT NULL PRIMARY KEY,
Jmeno VARCHAR2 (20) NOT NULL,
Prijmeni VARCHAR2 (20) NOT NULL,
RC CHAR (10) NOT NULL UNIQUE,
Tel CHAR (9) NOT NULL UNIQUE,
Email VARCHAR2 (50) NOT NULL UNIQUE,
Mesto VARCHAR2 (50) NOT NULL,
Ulice VARCHAR2 (50) NOT NULL,
CP VARCHAR2 (5) NOT NULL);

CREATE TABLE Pobocky (
idPobocky NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (50) NOT NULL,
Typ VARCHAR2 (50) NOT NULL,
Mesto VARCHAR2 (50) NOT NULL,
Ulice VARCHAR2 (50) NOT NULL,
CP VARCHAR2 (5) NOT NULL);

CREATE TABLE Kvalifikace (
idKvalifikace NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (50) NOT NULL,
Popis VARCHAR2 (1000) NOT NULL);

CREATE TABLE Pozice (
idPozice NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (50) NOT NULL,
Popis VARCHAR2 (1000) NOT NULL,
Volna CHAR (3) NOT NULL,
CHECK (Volna IN ('ano','ne')));

CREATE TABLE TypKomunikace (
idTypKomunikace NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (50) NOT NULL,
Popis VARCHAR2 (1000) NOT NULL);
```

```
CREATE TABLE Produkty    (
idProduktu NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (100) NOT NULL,
Cena NUMBER NOT NULL,
TypPlatby CHAR (11) NOT NULL,
Popis VARCHAR2 (1000) NOT NULL,
idKategorie NUMBER NOT NULL,
CHECK (Cena > 0 AND TypPlatby IN ('mesicni','jednorazova')),
FOREIGN KEY (idKategorie) REFERENCES Kategorie(idKategorie));

CREATE TABLE Zamestnanci    (
idZamestnance NUMBER NOT NULL PRIMARY KEY,
Jmeno VARCHAR2 (20) NOT NULL,
Prijmeni VARCHAR2 (20) NOT NULL,
RC CHAR (10) NOT NULL UNIQUE,
Vzdelani VARCHAR2 (10) NOT NULL,
Tel CHAR (9) NOT NULL UNIQUE,
Email VARCHAR2 (50) NOT NULL UNIQUE,
Mesto VARCHAR2 (50) NOT NULL,
Ulice VARCHAR2 (50) NOT NULL,
CP VARCHAR2 (5) NOT NULL,
Mzda NUMBER NOT NULL,
Dochazka NUMBER NOT NULL,
BankovniSpojeni CHAR (14),
idPozice      NUMBER NOT NULL,
idOddeleni NUMBER,
CHECK (Mzda > 0),
FOREIGN KEY (idPozice) REFERENCES Pozice(idPozice));

CREATE TABLE Oddeleni    (
idOddeleni NUMBER NOT NULL PRIMARY KEY,
Nazev VARCHAR2 (50) NOT NULL,
Popis VARCHAR2 (1000),
Tel CHAR (9) NOT NULL  ,
Email VARCHAR2 (50) NOT NULL,
idPobocky NUMBER NOT NULL,
idZamestnance NUMBER NOT NULL UNIQUE,
FOREIGN KEY (idPobocky) REFERENCES Pobocky(idPobocky),
FOREIGN KEY (idZamestnance) REFERENCES Zamestnanci(idZamestnance));

ALTER TABLE Zamestnanci
ADD FOREIGN KEY (idOddeleni) REFERENCES Oddeleni(idOddeleni);
```

```
CREATE TABLE Komunikace (
idKomunikace NUMBER NOT NULL PRIMARY KEY,
Ucel VARCHAR2 (50) NOT NULL,
Vysledek VARCHAR2 (50) NOT NULL,
Datum DATE NOT NULL,
Forma VARCHAR2 (20) NOT NULL,
idKlienta NUMBER NOT NULL,
idTypKomunikace NUMBER NOT NULL,
idZamestnance NUMBER NOT NULL,
FOREIGN KEY (idKlienta) REFERENCES Klienti(idKlienta),
FOREIGN KEY (idTypKomunikace) REFERENCES TypKomunikace(idTypKomunikace),
FOREIGN KEY (idZamestnanci) REFERENCES Zamestnanci(idZamestnance));

CREATE TABLE Smlouvy (
idSmlouvy NUMBER NOT NULL PRIMARY KEY,
Stav VARCHAR2 (20) NOT NULL,
DatumPodpisu DATE,
PlatnostDo DATE,
Komentar VARCHAR2 (1000),
idKlienta NUMBER NOT NULL,
idZamestnance NUMBER NOT NULL,
FOREIGN KEY (idKlienta) REFERENCES Klienti(idKlienta),
FOREIGN KEY (idZamestnance)REFERENCES Zamestnanci(idZamestnance));

CREATE TABLE Smlouva_obsahuje_Produkty (
idSmlouvy NUMBER NOT NULL,
idProduktu NUMBER NOT NULL,
FOREIGN KEY (idSmlouvy) REFERENCES Smlouvy(idSmlouvy),
FOREIGN KEY (idProduktu) REFERENCES Produkty(idProduktu));

CREATE TABLE Zamestnanec_ma_Kvalifikace (
idZamestnanci NUMBER NOT NULL,
idKvalifikace NUMBER NOT NULL,
FOREIGN KEY (idZamestnance) REFERENCES Zamestnanci(idZamestnance),
FOREIGN KEY (idKvalifikace) REFERENCES Kvalifikace(idKvalifikace));
```