Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství (PEF)

Bakalářská práce

Databázově koncipované zabezpečení provozu e-shopů

Pavel Jakl

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Pavel Jakl

Systémové inženýrství a informatika
Informatika

Název práce

Databázově koncipované zabezpečení provozu e-shopů

Název anglicky

Conceived database security of e-shops operation

Cíle práce

Bakalářská práce je zaměřena na problematiku databázově koncipovaného informačního zabezpečení provozu e-shopu. Náplní a účelem této práce je:
a) objasnit teoretické principy relačně databázové technologie v kontextu s problematikou zabezpečení provozu e_shopu,
b) zmapovat momentální stav této problematiky a vymezit její relevantnost včetně požadavků na ni kladených,
c) navrhnout možnosti přijatelných řešení těchto požadavků,
d) ověřit funkčnost navržených záležitostí,
e) ověřené záležitosti zobecnit pro další možná uplatnění.

Metodika

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

Závazný harmonogram:

Vymezení teoretických principů řešené problematiky, literární rešerše – do 5.9.2019: předmět 1. zápočtu z BP,

Zmapování současné situace řešené problematiky a navržení odpovídajícího řešení – do 20. 12. 2019,

Ověření navrženého řešení – do 31.1.2020: předmět 2. zápočtu z BP

Zobecnění navrhovaných záležitostí – do 10.3.2020: předmět 3. zápočtu z BP.

Doporučený rozsah práce

45-55 stran

Klíčová slova

Relačně db technologie, informační zabezpeční, provozování e-shopů. SQL, datová integrita

---

Doporučené zdroje informací

GROFF, James R. a Paul N. WEINBERG. SQL: kompletní průvodce. 2005. Brno: CP Books, 2005. Programování. ISBN 8025103692.

MORKES, David. Microsoft SQL Server 2000: tvorba, úprava a správa databází. Praha: Grada, 2004. Podrobný průvodce začínajícího uživatele. ISBN 8024707322.

URMAN, Scott, Ron HARDMAN a Michael MCLAUGHLIN. Oracle: programování v PL/SQL. Brno: Computer Press, 2008. Programování (Computer Press). ISBN 978-80-251-1870-2.

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

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

doc. Dr. Ing. Václav Vostrovský

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 9. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 9. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci " Databázově koncipované zabezpečení provozu e-shopů " jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2021 _____

Poděkování

Rád bych touto cestou poděkoval panu Doc. Ing. Václavu Vostrovskému, Ph.D. za odborné vedení bakalářské práce a za informace a rady spojené se zde probíranou tématikou.

Databázově koncipované zabezpečení provozu e-shopů

Abstrakt

Cílem této závěrečné práce je vytvořit obecný databázový model pro datovou evidenci e-shopu fiktivní firmy, která vstupuje na trh. Budou objasněny teoretické principy relačně databázové technologie, zabezpečení a požadavky na příslušný datový model.

Hlavním bodem pak bude vytvoření obecné zabezpečené databáze na základě dříve definovaných požadavků. K návrhu budou použity tři metody, první, relační vztahy, což znamená vztah mezi jednotlivými tabulkami v databázi. Následně datová integrita, kterou si vymezíme na referenční a doménovou. Jako poslední datová normalizace, kdy je zahrnuta atomicita, tedy dodržení určitých pravidel. Teprve poté přijde na řadu návrh zabezpečení.

Vytvořená databáze bude srovnána s těmi, které jsou vytvořeny automaticky redakčními systémy.

Navrhované řešení pomůže dosáhnout rychlejšího a bezpečnějšího chodu e-shopu a uživateli redakčního systému pomůže přiblížit problematiku tvorby databáze.

Klíčová slova: Relačně databázová technologie, informační zabezpečení, provozování e-shopů. SQL, datová integrita, normalizace dat, ORACLE

Conceived database security of e-shops operation

Abstract

The aim of this final thesis is to create a general database model for data registration of the e-shop of a fictitious company entering the market. The theoretical principles of relational database technology, security and requirements for the relevant data model will be clarified.

The main point will be the creation of a general secure database based on previously defined requirements. Three methods will be used for the design, the first, relational relationships, which means the relationship between the different tables in the database. Then data integrity, which we define as reference and domain. As the last data standardisation where atomicity is included, i.e. compliance with certain rules. Only then will the security proposal come into play.

The created database will be compared with those created automatically by editorial systems.

The proposed solution will help to achieve faster and safer running of the e-shop and will help the user of the editorial system to approximate the issue of database creation.

Keywords: Relational database technology, information security, operation of e-shops. SQL, data integrity, ORACLE, data normalization

Obsah

Seznam obrázků

Úvod

Tématem této závěrečné práce je navržení bezpečné a obecné databáze pro běžný chod e-shopu, vymezení kladů a záporů fungování komerční databáze vůči té, která je navržena na míru. Toto téma jsem si vybral z důvodu osobního zájmu o tuto oblast a zpracování informací z předmětu Databázové systémy.

Pokud je v dnešní době řeč o uchování a archivaci dat, téměř vždy se vybaví otázka, kde se tyto data ukládají. Ať už to byly v minulosti kartotéky, nebo organizované dokumenty, tak dnes jsou to databáze a databázová technologie obecně. Celá nynější společnost je postavena na informačních technologiích. Velké korporáty, střední a malé firmy, každý z těchto subjektů potřebuje archivovat svá data a pokud se rozhodnou pro elektronickou formu uchování dat, přichází zde na scénu databázový systém. Každý si podle svých potřeb a možností volí databázový produkt. Mezi známé firmy, které poskytují databázové produkty, patří například Oracle, Microsoft nebo IBM. I tyto firmy mají své podnikání založené na databázové technologii.

Po vytvoření databáze je vždy nutné klást velký důraz na bezpečnost a ochranu samotné databáze. Spousta serverů pracuje s databázemi, takže je nutné je zabezpečit jak zvenčí, tak i zevnitř firmy. Ochranu je nutné vykonat na několika úrovních. Tyto způsoby jsou běžně dostupné, ale bohužel někdy nevyužité. Jsou případy, kdy byla napadena rozsáhlá a „dobře zabezpečená“ databáze a byli vyneseny citlivá data klientů, nebo dokonce tajné vládní informace. V případě e-shopu jsou citlivá data zákazníků, ale také firmy. Další součástí je ochrana dat uvnitř firmy. Ne každý zaměstnanec má dostatečné pověření a oprávnění na to, aby mohl vidět, upravovat nebo mazat určitá data. K tomuto nám dokonale poslouží pohledy.

Zde jsou samotní zákazníci, kteří využívají výhod po přihlášení do e-shopu. Toto přihlášení musí být intuitivní a jednoduché, abychom zákazníka neodradili velkou složitostí registrace a login.

Aktuálnost tohoto tématu neztrácí na významu, ba naopak. Proto je zabezpečení datového modelu a jeho celá logika jedna z nejdůležitějších věcí při tvorbě jakékoli aplikace, e-shopu nebo online-katalogu.

V dnešní době totiž vzniká mnoho e-shopů a každý má jasný cíl a to, získat co nejvíce zákazníků. Aby stanoviska internetový obchod mohl dosáhnout, potřebuje dobrou a zabezpečenou databázi, která uchovává citlivé informace o zákaznících, ale i dodavatelích a celém chodu obchodu obecně.

1 Cíl práce a metodika

1.1 Cíl práce

Cílem této předkládané bakalářské práce je navržení a následné vytvoření funkční databáze pro fiktivní e-shop s důrazem na jeho rychlejší a bezpečnější chod. Dílčími cíli pak budou stručně a obecně objasnit teorii, která souvisí s problematikou zabezpečení dat. Na základě teoretické části pak zprostředkovat řešení, které bude brát v potaz všechny poznatky uvedené problematiky.

Navrhované řešení poté pomůže dosáhnout rychlejšího a bezpečnějšího chodu e-shopu, i za předpokladu, že e-shop nemusí dosahovat takové zabezpečovací úrovně jako například ČNB nebo informační systémy korporátních společností.

Navrhované řešení pomůže uživatelům, zejména pak těm, kteří využívají automaticky vytvořené databáze, lépe pochopit uložení a komunikaci dat v databázi a s tím související základy zabezpečení.

1.2 Metodika

1.2.1 Metody a techniky

- Shromáždění podkladů pro tvorbu teoretické části a vymezení metod, které budou využity při teoretické části.
    - Dostupné informační zdroje
    - Již vytvořená řešení v dané oblasti
- Analyzování nashromážděných zdrojů
    - Knižní publikace
    - Internetové zdroje
    - Vlastní zdroje
- Obecný popis problematiky, úvod do SQL jazyka
- Sepsání metod, které budou využity při tvorbě praktické části databáze
    - Relační vztahy
    - Datová integrita
    - Datová normalizace

- Zabezpečení databázového modelu
    - Uživatelé
    - Role
    - Pohledy
    - Práva
    - Profily

1.2.2 **Vlastní metodika**

Obsahem bude zmapování současné situace, což zahrnuje uvedení příkladů existujících e-shopů a jejich registrace, popřípadě přihlášení do zákaznického účtu. Dále také zmínění možností přihlášení a rozdíly mezi jednotlivými formami přihlášení.

Také zahrneme popis modelového e-shopu, následně vytvoření fiktivní společnosti, ke které se bude vztahovat tvorba databáze a také její popis. Dalším krokem je samotný návrh databáze, který zahrnuje vytvoření jednotlivých tabulek databáze, určení primárních a cizích klíčů a následné vytvoření relačního schématu databáze. K návrhu budou použity tři metody, první, relační vztahy, což znamená vztah mezi jednotlivými tabulkami v databázi. Následně datová integrita, kterou si vymezíme na referenční a doménovou. Referenční nejlépe vysvětlíme na příkladu, pokud smažeme jednoho uživatele z databáze, tak se smaže vše k němu navázané (anonymizují, null nebo se úplně odstraní). Doménová znamená, že má buňka určitá omezení, tedy pokud si příklad uvedeme na stipendium, které nemůže být záporné a zároveň vyšší než deset tisíc korun, bude buňka omezená (0;10000) Kč. Jako poslední je zde datová normalizace, kdy je zahrnuta atomicita, tedy, že musí být dodržena určitá pravidla. Opět nejlépe na příkladu, pokud bude ve formuláři kolonka „Jméno a příjmení“ je to rozdílné od dvou kolonek „Jméno“ a „Příjmení“, kdy v prvním případě právě není dodržena atomicita. Teprve poté přijde na řadu návrh zabezpečení. Ten zahrnuje vytvoření jednotlivých uživatelů, přidělení práv, vytvoření určitých rolí, specifických pohledů a vytvoření profilů.

V posledním kroku přichází na řadu samotné testování. V případě potřeby je tento krok opakován do bodu, kdy je databáze připravena a řádně otestována. Po úspěšném testovaní vzniká funkční datový model.

V závěru této práce pak shrneme všechny dílčí úkony procesu

2 Relační a databázové technologie

2.1 Historie databází

V roce 1965 se formovala konference o jazycích datových systému, kde vznikl výbor známy jako DBTG (Database Task Group), která měla za úkol standardizačním způsobem vytvořit koncepci databázového systému. [3]

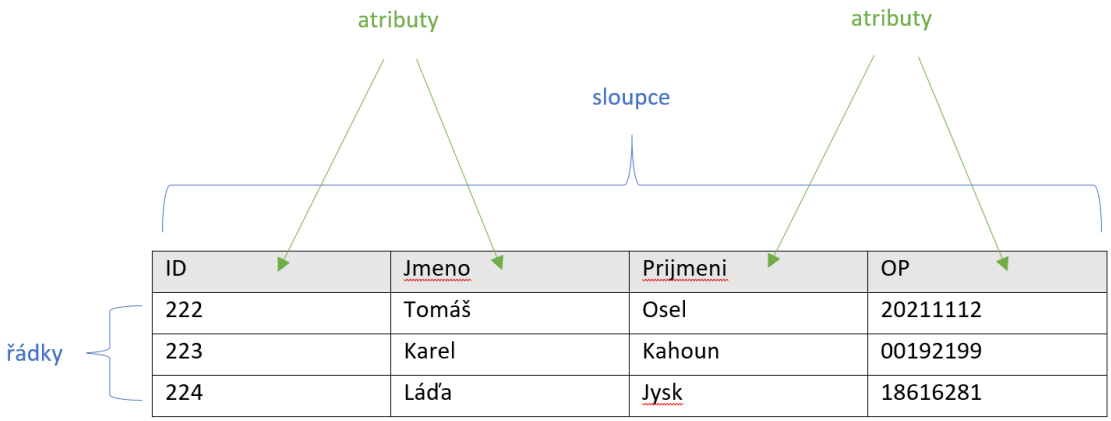
První SŘBD vznikli ke konci 60 letech a vycházeli ze dvou přístupů:

a) vzájemně propojené soubory dat (síťové a hierarchické databáze)
b) fyzicky nezávislé soubory dat (relační databáze) [6]

2.2 Relační databáze

Éra relačních databází začíná v roce 1970 článkem E. F. Codda v časopise Communications of ACM. Nejdříve šlo o teoreticky specifikovaný přístup, který vrátil data v izolovaných souborech, ale také pozvedl logickou úroveň pohledu na data. Uživatel viděl tyto data jako relace (tabulky) a dostal také k dispozici nástroje pro práci s těmito daty [3].

Relační databáze vychází z relačního modelu, který ukládá data do dvourozměrné tabulky, kde sloupce představují atributy tabulky a řádky samotné záznamy neboli fyzické entity. Atributy se volí tak, aby byly pro každou tabulku unikátní a v řádcích definujeme datový typ, čímž se specifikuje, jaká data budou v řádku uložena. [6]



ID	Jmeno	Prijmeni	OP
222	Tomáš	Osel	20211112
223	Karel	Kahoun	00192199
224	Láďa	Jysk	18616281

Obrázek 1 Obecný tvar relační tabulky – vlastní tvorba

Hlavní rysy relační databáze, které umožnují, že data mohou existovat nezávisle na své fyzické adrese jsou následující. Každý vztah je složen z uspořádaných entit, což jsou jednotlivé záznamy, a atributů, což jsou pole.

Samotné uspořádání záznamů v tabulce není vůbec podstatné, jelikož se rozlišuje podle pole, které je pro každý jednotlivý záznam v tabulce unikátní.

Relační databáze umožňují vytvářet vztahy mezi jednotlivými tabulkami v databázi, takže je možné přistupovat k datům z více tabulek, aniž by musela být předem definovaná nějaká hierarchie. Díky tomuto jsou relační databáze velmi flexibilní při zobrazování a třídění dat.

V důsledku toho, že relace jsou navzájem nezávislé, musí být vzhledem k značné obecnosti ve formulaci dotazů implementace relační databáze natolik dynamická, aby umožňovala dotazy zpracovávat efektivně. [3]

Dalo by se říct, že základními prvky, které tvoří databázi, jsou vztahy, omezení a pohledy.

Vztah znázorňuje propojení mezi tabulkami a jedná se o propojení primárního klíče jedné tabulky s cizím klíčem z druhé tabulky. Díky tomu je možné vybrat více tabulek a vybrat si z nich ty data, které se uživateli zdají nejvíce relevantní jeho hledání.

Omezení je pravidlo, které nám určuje, jaké hodnoty můžeme do objektu vložit. Po vytvoření určitého omezení nám do objektu nepůjdou vložit žádná jiná data než ta, která jsme specifikovali v omezení.

Pohled je možné si představit jako virtuální tabulku (tabulky), které jsou relevantní hledání uživatele. Mají tu výhodu, že slouží pouze k zobrazení informací, nikoli k zápisu. Administrátor může nastavit, které sloupce uživatel uvidí, ať už z důvodu nedostatečného oprávnění uživatele, nebo z důvodu toho, že dané informace k ničem nepotřebuje.

2.3 Databázová technologie

2.3.1 Základní pojmy databázové technologie

Pokud se shrne, co je to databázová technologie, tak lze říct, že je to unifikovaný soubor pojmů, prostředků a technik pro vytváření informačních systémů (**IS**) [7].

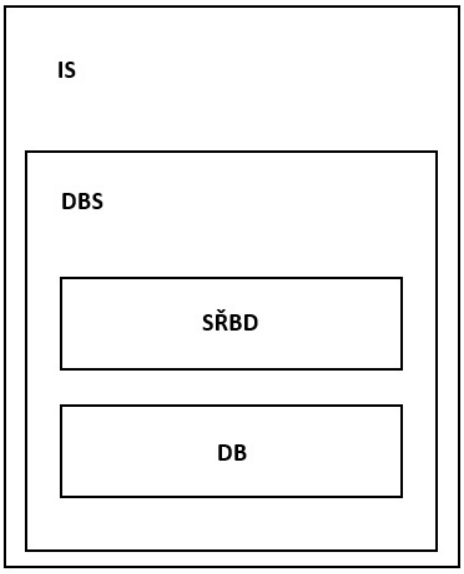
Na té nejnižší úrovni se IS a databáze představuje následovně:

1. Databáze
2. SŘBD
3. DBS
4. IS

Databáze slouží k ukládání a organizování dat a je řízena balíkem aplikačních programů neboli **SŘBD** (systém řízení báze dat), který většinou běží pod daným operačním systémem počítače. Současné SŘBD jsou z drtivé většiny založeny na transakčním zpracování dat. Pod tímto pojmem si můžeme představit posloupnost databázových operací. Databáze spolu se SŘBD tvoří tzv. **DBS** (databázový systém) [3]. Jejich vztah můžeme popsat následovně:

$$DB + SŘBD = DBS$$

Zde je vymezení pojmu **IS** (informační sytém), který využívá DBS buď přímo nebo pomocí nějakého dalšího aplikačního programu.



Obrázek 2 Schéma IS – vlastní tvorba

2.4 Relační vztahy mezi entitami

Pokud je řeč o relační databázové technologii, je nutné, aby se vytvořilo více tabulek, mezi kterými se následně provede relace neboli se definují relační vztahy mezi nimi. Základním vztahem je **klíč**.

2.4.1 Primární klíč

Primární klíč je speciální pole, které obsahuje unikátní hodnotu každého záznamu. Toto pole musí mít určitou hodnotu a nikde se v něm nemůže vyskytovat hodnota NULL. Při absenci primárního klíče nemůžeme uskutečnit relaci mezi tabulkami. Pokud se vybírá primární klíč, je důležité, aby hodnota, která se do tohoto pole vloží, skutečně plnila hodnotu primárního klíče, a ne žádnou jinou. [6]

2.4.2 **Cizí klíč**

Cizím klíčem v tabulce se rozumí to, že pokud nějaké tabulky budou potřebovat relaci jedna na druhou, tak vezmeme primární klíč z určité tabulky a vložíme ho do druhé tabulky právě pod pojmem cizí klíč, jelikož tato tabulka již obsahuje svůj unikátní klíč a v podstatě tabulky se tedy jedná o „cizí klíč“. [3]

Cizí klíč zároveň také slouží jako datová integrita. Jedná se o datovou integritu na základě vztahů, tudíž hodnota cizího klíče musí odpovídat hodnotě primárního klíče. Tímto se dá předcházet tzv. sirotčím záznamům. Z pohledu e-shopu, pokud by na objednávce chyběl nebo by nemohl být jednoznačně identifikován zákazník, danou objednávku není možno vykonat. Nebo například u přihlášení, pokud by se uživatel chtěl přihlásit a nebylo by zcela jasně identifikované jeho jméno, do e-shopu by se nepřihlásil. [6]

2.5 **Datová integrita**

V začátcích SQL byly specifikovány pouze dotazy NOT NULL a UNIQUE (SQL 86). Postupně se začalo rozšiřovat integritní omezení, a to na kontrolování hodnoty v řádku – CHECK, zadání implicitní hodnoty při vkládání řádku – DEFAULT a zadání referenční integrity pomocí cizích klíčů – FORIGN KEY a primárních klíčů – PRIMARY KEY. Standart SQL 92 všechny tyto možnosti zobecnil. [3]

Kontrola, zda nebyla porušena integritní omezení, se provádí při každé modifikaci. Je prováděna buď na konci SQL příkazu, nebo na konci transakce, jejíž je příkaz součástí.

Ke ztrátě integrity může dojít mnoha způsoby (INSERT, DELETE, UPDATE):

- Při vložení záznamu s neexistujícím produktem
- Hodnota v poli je neplatná, například pro cenu bude omylem zadán text
- Vytvoření objednávky se zbožím, které není na skladě
- Systémové a aplikační chyby
- Při špatném zvolení primárního klíče
- Výpadek paměti

2.5.1 **Referenční integrita**

Referenční integrita definuje jisté logické vztahy mezi dvěma tabulkami. Jedna tabulka je vždy hlavní a druhá je vedlejší. [3]

Vezměme si například tabulku zaměstnanec, kde se vyskytuje id_pozice a tabulku pracovní pozice. Každý zaměstnanec má svoji pracovní pozici, tudíž primární klíč zaměstnance bude shodný s cizím klíčem tabulky pracovní pozice.

Použitím této integrity tak nemůže dojít ke smazání záznamu pracovní pozice, která je přiřazena zaměstnanci nebo ji změnit.

Zde je uveden příklad zapsání referenční integrity:

```
FOREIGN KEY (id_pozice) REFERENCES prac_pozice
```

2.5.2 **Entitní integrita**

Při vytváření primárního klíče se musí brát ohled na to, aby každý vložený záznam byl unikátní entita a databáze jako celek stále zůstávala modelem vnějšího světa.

Pokud bude pro nějakou tabulku definovaný primární klíč, tak při každé změně, nebo pokusu o změnu zápisu databáze kontroluje, zda nedošlo k duplicitě primárního klíče. Pokud bude vložen nový řádek se stejným primárním klíčem, vyskočí chybové hlášení o duplicitě primárního klíče.

2.5.3 **Doménová integrita**

Na úrovni jednotlivých sloupců se omezuje datový typ a popřípadě i délka, které může maximálně dosáhnout (CHAR, VARCHAR, atd.…). Doména je množinou platných datových hodnot. Výhodou je, že pokud se změní typ, nemusí se to řešit u všech sloupců rozptýlených po celé databázi, ale tyto hodnoty jsou uloženy na jediném místě v databázi.

2.6 **Datová normalizace**

Při návrhu databázových tabulek je snaha o tzv. normalizaci, která by měla vést ke snadné a jednoduché práci s tabulkami. Odstraňuje redundanci dat a zjednodušuje zápis

tabulek. Datová normalizace má několik stupňů a obecně se chce dosáhnout stupně co nejvyššího.

2.6.1 **1 normální forma**

Tabulka splňuje 1. normální formu, jestliže všechny její sloupce jsou již dále nedělitelné, tzn. atomické.

2.6.2 **2 normální forma**

Tabulka splňuje 2. normální formu, jestli splňuje 1. normální formu a všechny její atributy jsou plně závislé na primárním klíči, kromě primárního klíče. Proto platí pouze pro více primárních klíčů. Pokud má tabulka jen jeden, je 2NF splněna automaticky.

2.6.3 **3 normální forma**

Tabulka splňuje třetí normální formu, pokud splňuje 1NF a 2NF a zároveň neobsahuje sloupce, které lze přenést do jiné tabulky. Jedná se o tranzitivní závislost

2.6.4 **Boyce-Coddova normální forma**

BCNF je vlastně původní znění 3NF jak ji autoři v roce 1974 publikovali. Někdy bývá označována jako 3,5NF. Jde o to, že v BCNF musí pravilo 3NF splňovat i hodnoty uvnitř složeného primárního klíče.

2.7 **Analýza zabezpečení databázového modelu**

V dnešní době je zabezpečení databáze jasnou prioritou pro všechny firmy. Každá firma se snaží ochránit si svá data a data svých zákazníku a z těchto důvodů má každá společnost jasně definovaná pravidla, jak s novými daty a s novými profily zacházet.

Databázi může používat více uživatelů. Z toho důvodu je nutné rozdělit tyto uživatele a přiřadit jim jednoznačná práva neboli role. Každý uživatel databáze pak má k dispozici

jen ty data, která mu jeho role umožňuje. Těmito opatřeními se pak dají eliminovat nechtěné nebo úmyslné chyby. Systém tak umožňuje manipulaci s databází v závislosti na tom, zda uživatel má právo na to, aby smazal nebo upravil požadovaný obsah. Každý uživatel má jednoznačný identifikátor, aby nedošlo k chybě při udělování. [6]

2.7.1 **Uživatelé**

Uživatele můžeme rozdělit do třech skupin:

- koncoví uživatelé
- objekty v databázi – aplikace
- správci

Každá z těchto skupin má určitá práva pro přístup do databáze. Například koncoví uživatelé mají omezený přístup. Pracují s daty, která jim byla přidělena a které potřebují pro svou práci.

Aplikace, popřípadě programátoři, které tyto aplikace napsali, aby usnadnili práci koncovým uživatelům.

Správci mají obvykle nejvyšší práva a starají se o samotnou databázi. Správci vytvářejí role a přidělují je uživatelům.

Nového uživatele vytvoříme pomocí příkazu:

```
CREATE USER Mary;
```

Pokud se vytvoří uživatel s heslem, příkaz bude následující:

```
CREATE USER Mary IDENTIFIED BY heslo;
```

Pokud se uživateli umožní změna hesla:

```
CREATE USER Mary IDENTIFIED BY heslo PASSWORD EXPIRE;
```

Dále se uživatel upravuje pomocí příkazu **ALTER USER**.

Pokud je třeba odstranit uživatele, použije se příkaz **DROP USER**.

2.7.2 Role

Každý podnik ke své databázi pouští své zaměstnance. Zaměstnanci mají různá oprávnění, a ta se často opakují. Kvůli tomu vznikly role, které uživatele se stejnými právy sdružují do skupin a ulehčují tím práci. Rozlišují se serverové a databázové role. [2]

2.7.2.1 Serverové role

Pomocí pevných serverových rolí se snadno uživatelům přidělí konkrétní právo globálního přístupu k databázi. [8]

2.7.2.2 Uživatelské databázové role

Uživatelské databázové role jsou efektivním řešením správy oprávnění pro jednotlivé uživatele. V rámci toho lze uživatele rozdělit do skupin podle jejich funkce. Je mnohem snazší vytvořit uživatelsky definovanou roli s názvem shop_manager a do ní přiřadit uživatele než všem uživatelům přiřazovat oprávnění. Oprávnění stačí přidělit definované roli [8]

Roli vytvoříme obecně pomocí příkazu:

```
CREATE ROLE <Role_Name>[Not Identified] | [Identified By <heslo> | Externally |globally ]
```

Pokud bude vytvořena role, která je chráněna přístupovým heslem, příkaz bude následující:

```
CREATE ROLE dw_manager IDENTIFIED BY warehouse;
```

2.7.3 Pohledy

Pohledy jsou virtuální tabulky implementující uživatelská schémata. Pohledy mohou být využity hlavně k dotazovaní, ale mohou být také aktualizovatelné. V SŘBD pohledy představují mechanismus, který vede ke zlepšení přístupu k datům.

Pro uživatele vypadá pohled jako skutečná tabulka, avšak v pohledu nejsou uloženy skutečné hodnoty.

Syntaxe pro vytvoření pohledu je následující:

```
CREATE VIEW jmeno_pohledu AS dotaz
```

Pro úspěšné vytvoření pohledu je třeba mít přístup ke všem tabulkám, na které odkazujeme.

Problém ovšem nastává při aktualizaci pohledů. Při aktualizaci základních tabulek takový problém není, pohledy se aktualizují tím, že se aktualizují data v základních tabulkách. Ovšem v tabulkách, které vzniknou spojením nevede k jednoznačně definovaným aktualizacím základních tabulek. [3]

2.7.4 **Práva**

Různí uživatelé mohou mít různá práva. Mohou mít právo zapisovat, ale už nemusí mít právo update nebo delete. Různým uživatelům jsou přiřazeny různé sady rolí.

Podle standardu jsou to následující práva:

- select
- update
- insert
- delete

Když nějaký uživatel vytvoří tabulku, tak se stává jejím vlastníkem a získává pro ni veškerá práva. Uživatel, který s touto tabulkou bude pracovat už musí mít práva přidělená, a to zajistíme pomocí příkazu GRANT. [6]

2.7.4.1 **Přidělení práv**

Práva určitému uživateli lze práva přidělit pomocí příkazu GRANT. Pokud má uživatel dostat všechny práva k tabulce, použijeme ALL PRIVILEGES. Pokud jsou třeba jen určitá práva, vypíšou se za GRANT a přidá se jasný identifikátor uživatele, kterému mají být práva přidělena. [6]

Syntaxe GRANT:

```
GRANT privilege-type ON [TABLE] {table-Name | view-Name}
TO grantees
```

2.7.4.2 **Odebrání práv**

Práva lze samozřejmě také odebrat. Slouží k tomu příkaz REVOKE. Příkaz má téměř stejnou strukturu jako GRANT, zase se vypíšou práva, která se mají odebrat a jednoznačný identifikátor uživatele. Odebrat lze jen některá nebo všechna práva. [6]

Syntaxe REVOKE:

```
REVOKE privilege-type ON [TABLE] {table-Name | view-Name}
FROM grantees
```

2.7.5 **Profily**

Uživatelský profil je sada databázových a systémových omezení, které jsou přiřazeny konkrétním uživatelům. Pomáhá také s omezením hesel. [10] Pokud v databázi není vytvořený žádný profil, bude využitý profil DEFAULT, který má neomezené prostředky pro všechny uživatele.

Aby byl vytvořený profil, je nutné pro něj mít oprávnění.

Syntaxe CREATE PROFILE:

```
CREATE PROFILE {name-of-profile} LIMIT {resource-
parametrs | passwords-parametrs}
```

Limity profilů se dělí na dvě skupiny, a to na parametry prostředků a parametry hesel. Každá skupina zabezpečí profil z jiného úhlu. Pro potřeby této práce bude více popsána část s parametry hesel.

Parametry hesel

FAILED_LOGIN_ATTEMPTS – počet po sobě jdoucích neúspěšných pokusů o přihlášení před uzamčením uživatele. Výchozí hodnota je 10x.

PASSWORD_LIFE_TIME – počet dní, po které může uživatel použít stejné heslo pro ověření. Výchozí hodnota je 180 dní.

PASSWORD_REUSE_TIME – počet dní, po které může uživatel znovu použít heslo.

PASSWORD_REUSE_MAX – počet změn hesel potřebných k opětovnému použití aktuálního hesla. Je třeba nastavit hodnoty parametrů password_reuse_time i password_reuse_max, aby se tyto parametry projevily.

PASSWORD_LOCK_TIME – počet dní, na které Oracle uzamkne účet po vyčerpání maximálních pokusů o přihlášení. Výchozí hodnota je 1 den.

PASSWORD_GRACE_TIME – počet dní, kdy je uživatel varován, že je dané heslo potřeba změnit a je odemčen přístup k účtu. Pokud se během této doby heslo nezmění, dojde k ukončení platnosti hesla a účet se uzavře. [10]

PASSWORD_VERIFY_FUNCTION – specifikace zabezpečení hesla, které bude použito. Možnost vytvoření vlastního skriptu nebo využití skriptu třetích stran. [11]

3 Jazyk SQL

SQL (Structured Query Language) je jazyk pro organizování, správu a získávání dat uložených v databázi. Samotný jazyk není řídícím systémem a ani samotným produktem, je integrovaný v každém databázovém systému a nesmí se opomenout fakt, že jazyk SQL pracuje pouze s jedním typem databází a tím jsou relační databáze. [1]

SQL je daleko více než jen dotazovací nástroj, protože tento jazyk lze také využít k řízení všech funkci, které databázový systém poskytuje uživatelům a to včetně:

- Definice dat
- Získávání dat
- Manipulace s daty
- Řízení postupu
- Sdílení dat
- Integrita dat

Jazyk SQL je tvořen „jádry“ a to DML, DDL, DCL, TCC. Příkazy, které tvoří DML (Data Mamipulation Language) jsou SELECT, INSERT, COMMIT, DELETE, UPDARE ROLLBACK a umožňují manipulovat s daty v databázi. Ale neumožňují vytvářet a odstraňovat tabulky, nebo sloupce. K tomu slouží DDL, jazyk pro definici dat.

DDL (Data Definition Language) je tvořen příkazy CREATE, DROP a ALTER. Umožňují vytvářet tabulky a sloupce, mazat je, měnit strukturu tabulky a mnoho dalších. [6]

DCL (Data Control Language) slouží pro přirazení práv uživatelům nebo také pro řízení transakcí a je tvořen příkazy GRANT, REVOKE, ALTER USER, DROP USER.

TCC (Transaction Control Commands) slouží pro řízení transakcí a je tvořena příkazy COMMIT, ROLLBACK a SAVEPOINT. [4]

3.1 SQL injection

Tento typ útoku je asi jeden z nejznámějších, ale dnes už dostatečně ošetřený. Ať už samotným vývojářem aplikace nebo se o to postarají hojně rozšířené frameworky. I když nikdy není nic stoprocentní. [12]

3.1.1 Popis útoku

SQL injection, je v podstatě o jednoduchá věc. Tento typ útoku napadá databázovou vrstvu vsunutím kódu do nezabezpečeného vstupu. Díky tomu může útočník získat velmi citlivá data z databáze, zjistit její strukturu nebo databázi poškodit smazáním některých záznamů. [4]

SQL injection je možno rozdělit do několika skupin

- spojení více tabulek
- obcházení aplikační logiky
- pozměnění databáze
- získání informací o databázi

Spojení více tabulek

Tento typ se využívá, pokud je výsledek dotazu vracen přímo do prezenční vrstvy. Zde se používá příkaz **UNION**, který umožní spojit více příkazů **SELECT** dohromady. Tento ty se používá ve spojení URL stránky.

Obcházení aplikační logiky

Při tomto typu útoku se vkládá kód do inputu aplikace. Pokud máme například jednoduchý přihlašovací formulář a vložíme do inputu **name** například ' OR admin = 1--, tak se stane, že díky apostrofu se řetězec ukončí u sloupce name a vloží se podmínka. Dvě pomlčky nám zaručí to, že cokoli za nimi bude bráno jako komentář. [12]

4 Navrhované řešení

V současné době vzniká mnoho e-shopů a tato služba už není podmíněna velkou investicí do samotného produktu, ale naopak, stává se čím dál dostupnější pro každého, kdo by chtěl začít podnikat právě skrze e-shop.

Dnes existují platformy, kde si může kdokoli, za minimální náklady, vytvořit e-shop během pár dnů a nepotřebuje znát téměř žádné informace o programování, aplikační logice ani o samotné databázi. To vše vyřeší platforma, na které e-shop vytváříme.

Tady nastává možná nejzásadnější problém, a to ten, že databáze je tvořena automaticky. Sice ji tvořil tým vývojářů, kteří se starají o její chod a aktuálnost, ale jelikož je databáze obecná, nemůže nikdy uspokojit všechny zákazníky, kteří ji využívají a tyto open source databáze jsou také velice náchylné k napadnutí.

Je tu také další limitující věc. Jelikož tyto systémy podporují pluginy, které jsou uloženy v databázi, nastává problém. Při použití více pluginů se z databáze stává sněhová koule, která na sebe nabaluje stále nové a nové tabulky, a klesá tak její rychlost i bezpečnost.

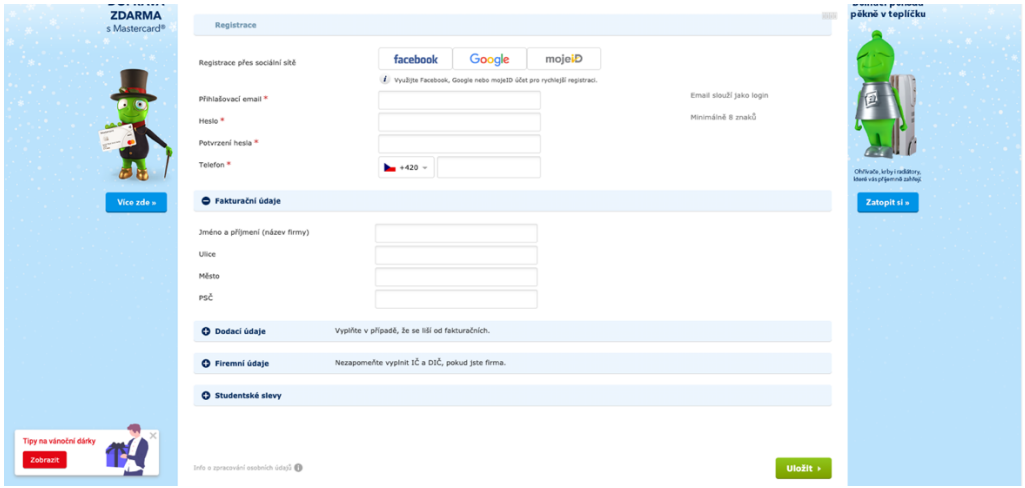
Databáze je srdcem každé aplikace nebo e-shopu a je tedy třeba ji nejen správně navrhnout, ale také ji správně udržovat a mít přehled o každé jedinečné tabulce, která byla vytvořena, a hlavně za jakým účelem.

Navrhované řešení bude odlišné od těchto typů databází z pohledu zabezpečení na úrovni samotné databáze a bude tvořené na konkrétní potřebu e-shopu.

Každý, kdo využívá databázi by si měl v první řadě určit, jaká data budou ukládána a podle tohoto faktu by měl učinit rozhodnutí, jak stavět zabezpečení a jak spravovat přístup ostatních do databáze.

4.1 Registrace

Dnešní registrace už nevyžadují pevné vyplnění údajů, jako tomu bylo kdysi, ale systémy už komunikují s třetími stranami, jako je Google, Facebook



Obrázek 3 Přihlášení do obchodu Alza – screenshot [13]

nebo moje ID a díky této funkci nemusíme znovu vyplňovat údaje, ale do databáze se uloží údaje, které jsou spojené s danou službou. Je to jistá výhoda používání těchto platforem, ale není podmínkou a uživatel tak může vyplnit prázdnou registraci.

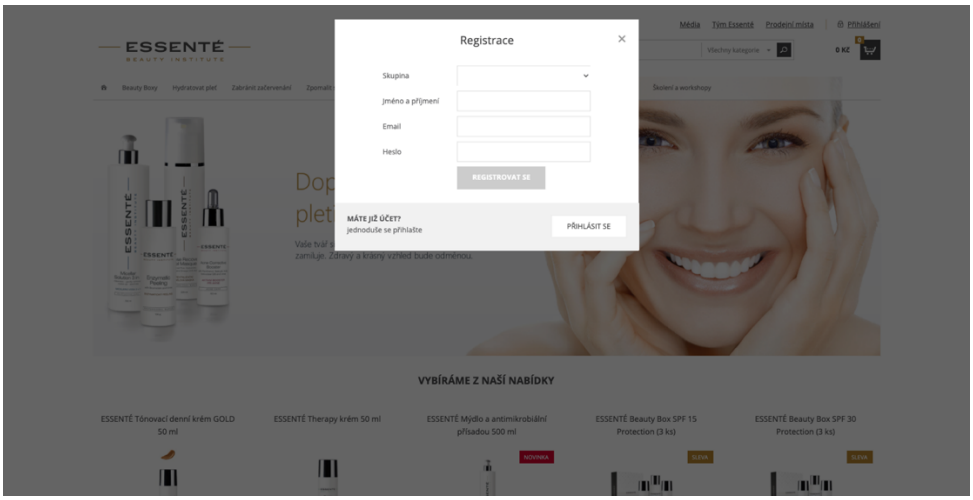
Jak si můžeme všimnout na obrázku 4, tak při registraci máme rozdělené sekce. Povinná sekce obsahuje klasické informace, jako nick, dvoufázové ověření hesla a email. Pro samotnou registraci nám to stačí, ale pro využití všech funkcí, které nám systém nabízí, musíme stejně vyplnit fakturační údaje, údaje dodací atd.

Jednotlivé inputy mají svou formu, která určuje, co bude inputem povoleno k odeslání. Například pole email musí obsahovat znak @, pole Telefon zase chce na svém inputu čísla. Díky tomuto je snazší udržet v databázi řád a jednotlivé sloupce tabulky tak budou obsahovat to, co se od nich očekává.

Dalším a jedním z těch hlavních je heslo, které má také specificky požadovanou formu. Je to jeho délka, může obsahovat speciální znaky a číslice. Při ukládání hesla do databáze pak projde ještě šifrováním, jako je například **MD5**, které vytvoří **HASH** hesla, a to se následně uloží do databáze.

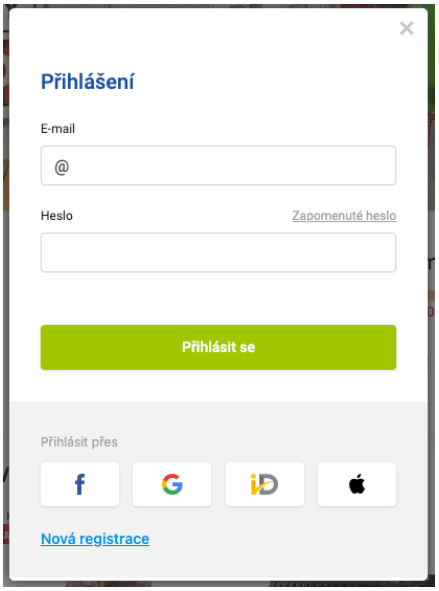
Mimo velkých hráčů, jako je Alza, Mall, CZC a další jsou ještě klasické registrační stránky. Na obrázku 5 můžeme vidět registraci, které využívá jména, hesla a emailu, ale

také nabízí možnost registrace podnikatele, přičemž se takovému zákazníkovi, který si založí firemní účet, budou zobrazovat zvýhodněné ceny.



Obrázek 4 ESSENTE registrace – screenshot [14]

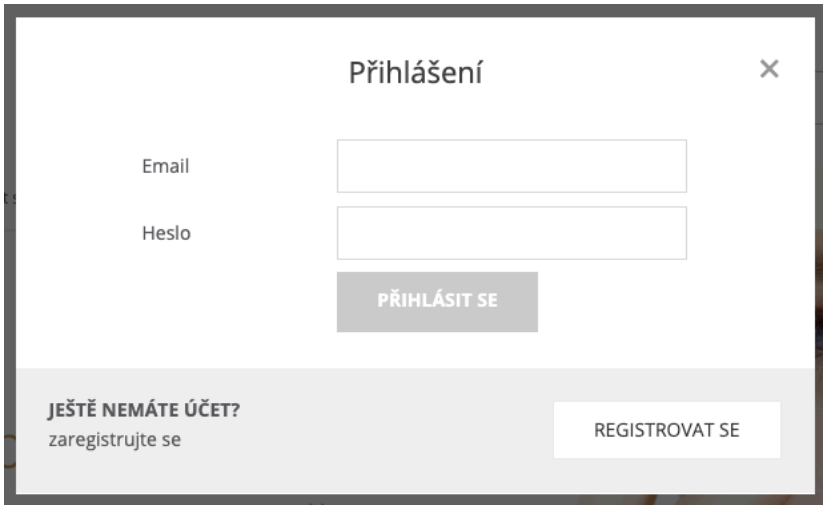
4.2 Přihlášení



Obrázek 5 Přihlášení obchodu Alza – screenshot [13]

Přihlášení do e-shopu je pak otázka chvíle. Lze se přihlásit přes již zmíněné služby třetích stran nebo klasicky přes vyplnění jména a hesla. Dnes mají již vyhledávače autocomplete a pokud máme uložená hesla, tak nám automaticky nabídne vyplnění emailu a s ním spojené heslo pro určitou stránku. Některé e-shopy mají i funkci zapamatovat si přihlášení, takže zákazník bude automaticky přihlášen.

Přihlašovací forma je drtivou většinou tvořena inputy **e-mail** a **heslo**. Jelikož se často stává, že zákazník zapomene své heslo, může si požádat o heslo dočasné, které si následně na unikátní url adrese změní. Tady je velmi důležitý fakt, že povinný údaj je právě email, jelikož kdyby email nebyl povinný a zákazník zapomněl své přihlašovací údaje, nemohl by si u svého účtu požádat o obnovení hesla, a tudíž by o svůj zákaznický účet přisel.



Obrázek 6 Přihlášení obchodu Essenté – screenshot [14]

Velkou výhodou uživatelkých účtů je, jak už zaznělo, možnost nakupování jako podnikatel, ale také jako student. Řada e-shopů nabízí studenstké slevy, například pomocí ISIC, a další výhody. Další velkou výhodou je také to, že je jasný přehled o všech objednávkách a také o jejich stavu.

4.3 Vytvoření databáze pro fiktivní e-shop

Databáze pro tento e-shop bude vytvořena z informací, které jsou popsány v teoretické části práce. Databáze bude postavena nad smyšleným internetovým obchodem.

Bude modelovat pouze základní princip fungování databáze, jelikož celý projekt by byl mnohem rozsáhlejší.

4.3.1 CartoonStore

Cartoonstore je internetový obchod, který se zabývá prodejem artiklů ze světa komiksů a kreslených seriálů. Začal jen jako malý blog, na kterém jeho majitel, nadšenec světa fantasy a komiksů, psal novinky z toho světa.

Po nějaké době si získal menší, ale pevnou základnu lidí, kteří pravidelně navštěvovali jeho web a on začal přemýšlet, že by mohl dělat to, co ho baví a zároveň si i něco vydělat. Proto se rozhodl, že si založí malý e-shop s komiksovou tématikou a prováže tak zábavu s nákupem.

Po úspěšném založení e-shopu na redakčním systému WordPress mu e-shop fungoval přesně podle očekávání. Propojil si svůj web s analytikou, a tak mohl sledovat, kolik lidí si nakoupí a kolik si jich chce číst jeho články.

Jelikož mu e-shop začal vydělávat, začal přemýšlet, že by si otevřel kamennou prodejnu a splnil si tak svůj sen. Za nějaký čas toto rozhodnutí zrealizoval a výsledky se pomalu dostávali. Velká část lidí, která nakupovala na e-shopu, chodila i do prodejny.

Jak podnikání rostlo, přibývali prodejny a zákazníci, přišel čas, aby se dostal na lepší úroveň i jeho e-shop. Nechal si tedy vytvořit e-shop na míru, který bude stabilní, rychlý a dobře zabezpečený a zde přichází základní kámen, a to je databáze, nad kterou bude e-shop postavený a je nutné ji dobře zabezpečit.

4.3.2 **Databáze e-shopu**

Databáze bude uchovávat informace o produktech e-shopu a zda jsou skladem, pak také informace o zákaznících a objednávkách. Bude zde zahrnuta i sleva pro zákazníky, kteří nakupují pravidelně.

Začněme tabulkou **users**. Tato tabulka obsahuje všechny uživatele v systému. Daná entita má dané atributy. Primárním klíčem zde bude user_id. Podle role se pak rozliší, jaké má uživatel práva.

```
CREATE TABLE users
(
  id INT NOT NULL,
  name CHAR(40) NOT NULL,
  password CHAR(30) NOT NULL,
  role INT NOT NULL,
  first_name CHAR(40) NOT NULL,
  last_name CHAR(40) NOT NULL,
  registration_token VARCHAR(191) NOT NULL,
  phone CHAR(13) NOT NULL,
```

```
  PRIMARY KEY (id)
);
```

Přesuneme se k entitě **products**. Zde se budou ukládat data o názvu, ceně, dostupnosti ve skladu, viditelnosti na stránce, textu a perexu a dalších atributů. Zde bude primární klíč product_id.

```
CREATE TABLE products
(
  id INT NOT NULL,
  name CHAR NOT NULL,
  slug CHAR NOT NULL,
  image_main VARCHAR(50) NOT NULL,
  cart_image VARCHAR(50) NOT NULL,
  perex VARCHAR(130) NOT NULL,
  text VARCHAR(191) NOT NULL,
  price INT (6) CHECK (price BETWEEN 0 and 100000),
  stock Boolean NOT NULL,
  published Boolean NOT NULL,
  stock_value INT NOT NULL,
  user_created_id INT NOT NULL,
  user_updated_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (user_created_id) REFERENCES users(id),
  FOREIGN KEY (user_updated_id) REFERENCES users(id)
);
```

Další entitou jsou **categories**. U kategorií se bude ukládat název kategorie a spojí se ještě s tabulkou Users, jelikož kategorie musel nějaký uživatel vytvořit.

```
CREATE TABLE categories
(
  id INT NOT NULL,
  name CHAR NOT NULL,
  user_created_id INT NOT NULL,
  user_updated_id INT NOT NULL,
```

```
  PRIMARY KEY (id),
  FOREIGN KEY (user_created_id) REFERENCES users(id),
  FOREIGN KEY (user_updated_id) REFERENCES users(id)
);
```

Vytvoření vazebné tabulky **products_has_category** mezi **users** a **categories**. Tato tabulka nám spojí **products** a **categories**.

```
CREATE TABLE product_has_category
(
  id INT NOT NULL,
  category_id INT NOT NULL,
  product_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (category_id) REFERENCES categories(id),
  FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Další tabulkou je **customers**. Tabulka customers bude uchovávat informace o zákaznících, kteří si vytvořili zákaznický účet, takže typicky se uloží jméno, příjmení, fakturační a dodací adresy, zákaznické karty nebo určitý druh slev.

```
CREATE TABLE customers
(
  id INT NOT NULL,
  firstname CHAR NOT NULL,
  lastname CHAR NOT NULL,
  billing_firstname VARCHAR(40) NOT NULL,
  billig_lastname VARCHAR(40) NOT NULL,
  billing_phone CHAR NOT NULL,
  billing_email CHAR NOT NULL,
  billing_street CHAR NOT NULL,
  billing_city VARCHAR(80) NOT NULL,
  billing_zip_code CHAR NOT NULL,
```

```
  delivery_firstname VARCHAR(40) NOT NULL,
  delivery_lastname VARCHAR(40) NOT NULL,
  delivery_company VARCHAR(50) NOT NULL,
  delivery_street VARCHAR(30) NOT NULL,
  delivery_city VARCHAR(40) NOT NULL,
  delivery_zip_code CHAR NOT NULL,
  delivery_address CHAR NOT NULL,
  company_name VARCHAR(50) NOT NULL,
  company_ico CHAR NOT NULL,
  company_dic CHAR NOT NULL,
  company_address CHAR NOT NULL,
  company VARCHAR(50) NOT NULL,
  customer_card_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (customer_card_id) REFERENCES
customer_cards(id)
);
```

Zákaznické karty poskytují zákazníkům čerpání bonusů za nákup. Každá karta náleží právě jednomu zákazníkovi a jeden zákazník má jednu kartu.

```
CREATE TABLE customer_cards
(
  id INT NOT NULL,
  price_group INT NOT NULL,
  discount INT NOT NULL,
  PRIMARY KEY (id)
);
```

Pro ukládání vybraného zboží do košíků poslouží tabulka **cart**. Eviduje se id produktu, zákazníka a množství objednaného zboží.

```
CREATE TABLE cart
(
  id INT NOT NULL,
```

```
  quantity INT NOT NULL,
  customer_id INT NOT NULL,
  product_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (customer_id) REFERENCES customers(id),
  FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Veškerá data o objednávce shrne tabulka **orders**. Zde se ukládají informace o dodaní, o fakturaci, vyzvednutí a platbě. Pokud je registrovaný zákazník, informace se uloží z tabulky customers pomocí cizího klíče.

```
CREATE TABLE orders
(
  id INT NOT NULL,
  billing_firstname VARCHAR(40) NOT NULL,
  billing_lastname VARCHAR(40) NOT NULL,
  billing_phone CHAR NOT NULL,
  billing_email CHAR NOT NULL,
  billing_street CHAR NOT NULL,
  billing_city CHAR NOT NULL,
  billing_zip_code INT NOT NULL,
  delivery_firstname VARCHAR(40) NOT NULL,
  delivery_lastname VARCHAR(40) NOT NULL,
  delivery_company VARCHAR(50) NOT NULL,
  delivery_street CHAR NOT NULL,
  delivery_city CHAR NOT NULL,
  delivery_zip_code CHAR NOT NULL,
  comapny_name VARCHAR(50) NOT NULL,
  comapny_ico CHAR NOT NULL,
  comapny_dic CHAR NOT NULL,
  note VARCHAR NOT NULL,
  customer_id INT NOT NULL,
```

```
    transport_method_id INT NOT NULL,
    payment_id INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (customer_id) REFERENCES customers(id),
    FOREIGN KEY (transport_method_id) REFERENCES
transport_methods(id),
    FOREIGN KEY (payment_id) REFERENCES paymants(id)
);
```

Tabulka **transport_methods** uchovává druhy dopravy. S tabulkou orders je spojena pomocí cizího klíče.

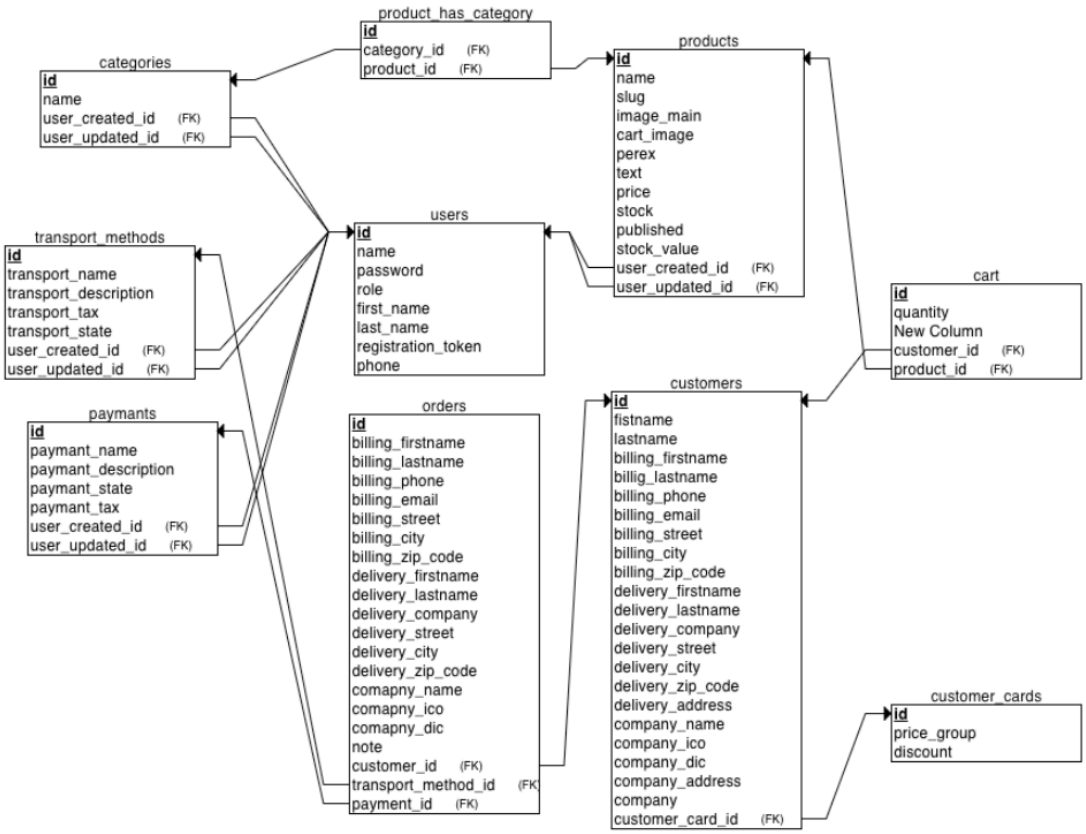
```
CREATE TABLE transport_methods
(
  id INT NOT NULL,
  transport_name CHAR NOT NULL,
  transport_description VARCHAR(100) NOT NULL,
  transport_tax INT NOT NULL,
  transport_state INT NOT NULL,
  user_created_id INT NOT NULL,
  user_updated_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (user_created_id) REFERENCES users(id),
  FOREIGN KEY (user_updated_id) REFERENCES users(id)
);
```

Způsoby platby jsou evidovány v tabulce **payments**. Různé druhy dopravy mají různě odlišení poplatky za jejich poskytování. Spadá sem tradiční platby na dobírku, vyzvednutí na prodejně a hlavní platební kanál, a to platba kartou jako služba třetí strany, se kterou aplikace komunikuje přes API.

```
CREATE TABLE paymants
(
  id INT NOT NULL,
```

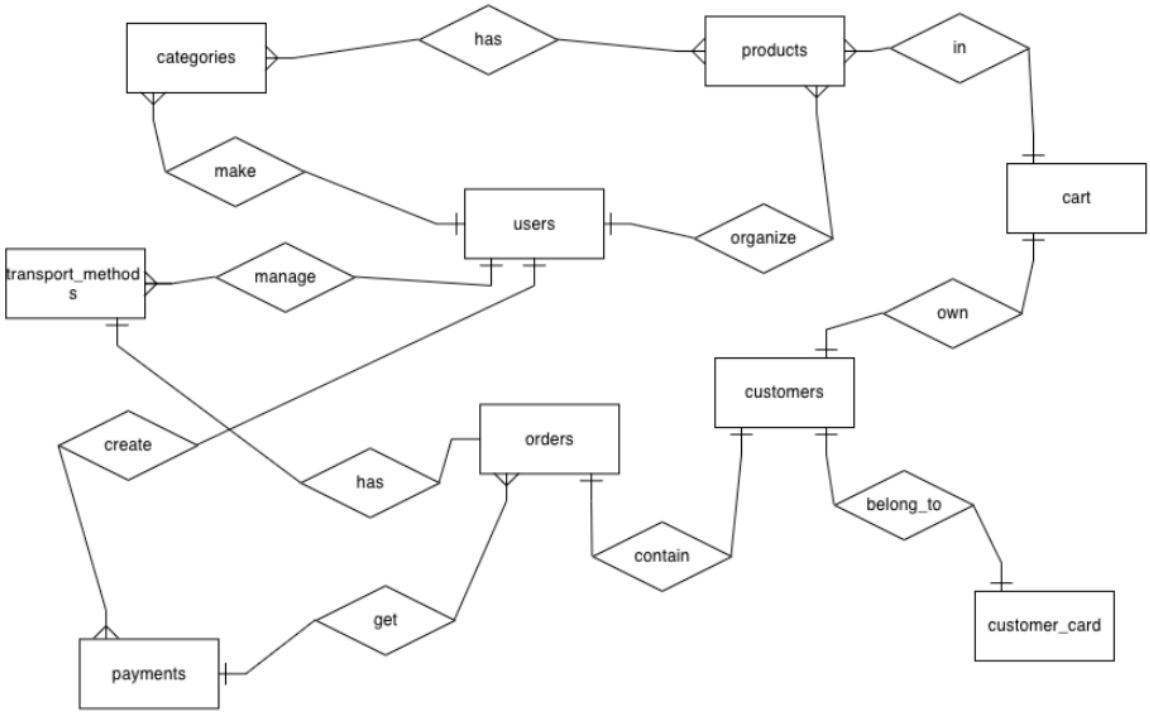
```
    paymant_name CHAR NOT NULL,
    paymant_description VARCHAR(100) NOT NULL,
    paymant_state INT NOT NULL,
    paymant_tax INT NOT NULL,
    user_created_id INT NOT NULL,
    user_updated_id INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (user_created_id) REFERENCES users(id),
    FOREIGN KEY (user_updated_id) REFERENCES users(id)
);
```

4.3.3 Relační schéma databáze



Obrázek 7 Relační schéma databáze – vlastní tvorba

4.3.4 Kardinalita vztahů v databázi



Obrázek 8 Kardinalita vztahů

4.4 Zabezpečení

Při takto vytvořené databázi se může začít řešit její zabezpečení. V e-shopu bude logicky vystupovat více osob, kde každý má různá práva a různé kompetence v hierarchii systému. Například editor produktů bude jen těžko měnit fakturační údaje firmy a dodávky zboží. Na druhou stranu fakturační oddělení zase potřebuje přístup k detailům objednávek a zákazníků pro kompletaci účetnictví.

Takže na základě nastavené hierarchie kompetencí můžeme vytvářet role a zobrazovat jednotlivým uživatelům tu část databáze, do které mají povoleno nahlédnout.

4.4.1 Uživatelé

Aby mohl v e-shopu vzniknout jakýkoli produkt, kategorie, objednávka nebo cokoli dalšího, je zapotřebí někoho, kdo tyto data vytvoří – uživatel. Když na začátku pomineme, jaký uživatel má jaké práva a pohledy, je nutné, aby vůbec uživatel vznikl.

Pro vytvoření uživatele se využije příkazu:

```
CREATE USER name IDENTIFIED BY name_password;
```

Tento příkaz vytvoří uživatele s určitým jménem a bude ověřován pomocí hesla.

Pro potřeby obchodu se vytvoří tedy 3 pozice a to administrátor, který bude mít dohled nad celým systémem a bude mít přístup do celé databáze. Dále manažer obchodu, který bude mít přístup k dodavatelům a k interní databázi zákazníků a objednávek a editor produktů, který bude mít přístup pouze do produktů.

Vytvoření všech tří uživatelů bude vypadat následovně:

Administrátor

```
CREATE USER Petr IDENTIFIED BY Ajkbd8bsd009;
```

Manažer obchodu

```
CREATE USER Jan IDENTIFIED BY pjPad8bsd192;
```

Editor

```
CREATE USER Ignác IDENTIFIED BY asjkbOVBsd0112;
```

Pokud se při vytvoření uživatele generuje náhodné heslo a je žádoucí, aby si jej uživatel změnil po přihlášení, této operace se docílí pomocí **PASSWORD EXPIRE**.
Takže příkaz, který by toto zajišťoval by byl následující:

```
CREATE USER Josef IDENTIFIED BY oqinwW99WnxoW PASSWORD
EXPIRE;
```

Nastala situace, že si pan Ignác zapomněl uložit přihlašovací údaje a je nutné aktualizovat jeho heslo. Pro tuto situaci se využívá **ALTER**

```
ALTER USER Ignác IDENTIFED BY new_password;
```

Uživatelé v obchodu

Ve stejném duchu fungují i registrace zákazníků do e-shopu. Uživatel si vytvoří svůj zákaznický účet, ve kterém specifikuje svoje přihlašovací jméno a email. Pokud je

uvedeno, že nový uživatel si musí po přihlášení změnit heslo, již je známo jak, tak se po přihlášení zobrazí doporučení ke změně hesla.
Stejně tak při zapomenutí přihlašovacích údajů. Proto je daný uživatelský profil spojený právě s jedním e-mailem. Při zapomenutí hesla je na e-mail odeslán e-mail s pokyny pro obnovu hesla a po jeho provedení se v databázi odehraje scénář ALTER USER. Zákazník ve svém profilu uvádí nejen své jméno a e-mail, ale také dodací a fakturační údaje, popřípadě pokud je podnikatel, udává informace jako IČ a DIČ.
Pokud je třeba uživatele nebo zákazníka ze systému odstranit, uděláme to pomocí příkazu **DROP**. Takže příkaz, který zajistí smazání uživatele či zákazníka ze systému bude vypadat takto:

```
DROP USER Josef;
```

4.4.2 **Práva**

Po specifikování uživatelů a vytvoření podnikové hierarchie se jako další v pořadí musí vytvořit práva, kterými jednotlivý uživatelé systému budou disponovat.

Pro vytvoření práv jednotlivým uživatelům se používá příkaz **GRANT**.

Uživatel Ignác je editorem v obchodu, tudíž se musí zamezit tomu, aby se Ignác nedostal do částí databáze, ke které nemá oprávnění. Příkaz pro přidělení práv Ignácovi by vypadal následovně.

```
GRANT ALL ON products, categories TO Ignác;
```

Nyní je jasně specifikováno, jaké práva může Ignác v rámci podnikové databáze využívat a na jakých tabulkách.

Pokud by se vedení rozhodlo, že Ignác již není spolehlivý a bude z editora sesazen pouze na kontrolora obsahu, změní se jeho práva pomocí příkazu **REVOKE**.

```
REVOKE UPDATE, INSERT, DELETE ON products, categories
FROM Ignác;
```

Je situace, kdy se žádná práva přidělovat nemusí, a to, když je tabulka vytvořena. Vlastník tabulky (ten, kdo specifikoval příkaz CREATE TABEL) má automaticky všechna

práva a může s tabulkou pracovat v plném rozsahu. Může nastavit práva ostatním uživatelům.

4.4.3 **Role**

Role jsou jedním z hlavních prvků zabezpečení databáze. Jedná se o skupiny se stejnými pravomocemi. Při vytvoření dané skupiny je nutné této skupině přidělit práva a následně přidat uživatele do této skupiny.

Internetový obchod se rozrostl a již má několik editorů. Místo toho, aby se každému jednotlivému editorovi přidělovala samostatně práva, vytvoří se role Editor, která už bude mít všechna potřebná práva specifikována.

```
CREATE ROLE editor;
GRANT ALL ON products, categories TO editor;
```

Takto vytvořenou roli se specifikovanými právy pak přidělíme uživatelům.

```
GRANT editor TO Ignác;
```

V hierarchii obchodu bude rolí hned několik, jelikož, jak je výše uvedeno, je zde nějaký administrátor, který se stará o chod a může pracovat s databází v plném rozsahu, manažer obchodu, který bude mít přístup jak k produktům, tak zároveň k fakturačním údajům, k formě dopravy a určitě také k datům dodavatelských firem. Bude tu účetní, která bude mít přístup hlavně k fakturám přijatých i vydaných.

Zde se to může lišit, jelikož firma může použít externí účetní systém a data se pak z e-shopu pomocí API requestu posílají do externího systému, kde s nimi dál pracují. Tato implementace může být jen částečná nebo úplná, včetně skladového hospodářství a celkové obsluhy internetového obchodu. Ale zde zase narážíme na to, že tento externí systém běží nad nějakou databází, kde jsou také jasně určené role a kdo má přístup do jaké části systému.

Příklad vytvoření role pro manažery obchodu.

```
CREATE ROLE shop_manager;
GRANT ALL ON transport_methods, paymants, categories,
products TO shop_manager;
GRANT SELECT, UPDATE ON orders, cart TO shop_manager;
```

```
GRANT SELECT ON customers, orders, cart TO shop_manager;
GRANT shop_manager TO Jan;
```

Určité role mohou být ještě zabezpečeny heslem. Mohlo by se to zdát už lehce zbytečné, ale když nastane situace, že manažer obchodu nebude moci v danou chvíli pracovat, tak většinou má svého zástupce. Zástupce má logicky nižší práva než manažer, ale zná práci manažera, takže v krajním případě může zastoupit a musí právě heslo znát, aby mohl používat danou roli.

Již vytvořený pohled by tedy mohl být modifikovaný následujícím způsobem.

```
CREATE ROLE shop_manager IDENTIFIED BY A32ds52aSDQ32a;
```

Po vytvoření rolí se může databáze považovat za více zabezpečenou. Jelikož je jasně specifikováno, kdo má jaké role, jaké mají role práva. Pro přehlednost a funkčnost podniku je to ohromná výhoda, jelikož její zaměstnanci jsou rozděleni podle svého zaměření do konkrétních skupin (rolí).

Při situaci, kdy bude mít modelová firma řádově stovky zaměstnanců se už projeví nezbytnost rolí, jelikož udržovat pod kontrolou jednotlivé uživatele a jejich práva bez konkrétního zařazení do skupiny je časově i procesně náročné.

4.4.4 Pohledy

K dalšímu popisu dat v podniku slouží pohledy. Pohled je tabulka, která však není fyzicky deklarovaná, ale vzniká při použití specifického pohledu, tedy provedením příkazu SELECT.

Příkaz, který deklaruje nový pohled vypadá následovně:

```
CREATE VIEW <jmeno-pohledu> [seznam-sloupcu] AS <SELECT>
```

V podnikovém systému se může jednat o vytvoření pohledů například ke všem vyřízeným objednávkám, ke kterým bude mít přístup manažer obchodu. Vytvoří se tedy nový pohled, příkaz SELECT, který vybere ze všech objednávek ty, které jsou vyřízené.

Příkaz bude vypadat takto:

```
CREATE VIEW completed-order AS SELECT * FROM orders WHERE
state=5;
```

Pokud se pohled vytvoří, lze už bez problémů vypsat data, které v sobě pohled seskupil.

Příkaz, který vypíše všechny informace o vyřízených objednávkách pomocí pohledu bude vypadat následovně:

```
SELECT * FROM completed-orders;
```

Vzhledem ke skutečnosti, že na e-shopu máme několik možností platby a když pomineme platby online, které se řeší ihned, tak je zde i možnost platby na bankovní účet.

Po vybrání této možnosti se při souhrnu objednávky objeví buď číslo účtu nebo QR kód a zákazník odešle požadovanou částku na účet. Pro lehčí orientaci se vytvoří pohled, který bude vybírat id objednávky a zákazníky, kteří zvolili platbu na účet, a ještě tuto platbu neprovedli.

Pohled bude vypadat následovně:

```
CREATE VIEW payment-on-account AS SELECT id, customer_id, paymentstatus, payment, firstname, lastname WHERE orders.customer_id = customers.id AND payment = 2 AND paymentstatus = 1 ORDER BY id DESC;
```

Po vypsání pohledu příkaz zobrazí všechny objednávky, u kterých je zvolena platba na účet a nejsou ještě zaplaceny.

Pro úplnost by zde šlo navázat na data splatnosti a pomocí triggeru udělat oznámení o konci splatnosti u konkrétní objednávky, ale v rozsahu této práce je to řešeno přes tento konkrétní pohledy.

4.4.5 **Profily**

Nejprve je nutné říct, že jednomu konkrétnímu uživateli, lze přiřadit pouze jeden profil. Jestli existuje uživatel, který má již profil přiřazený a bude mu i tak přiřazen jiný profil, ten starý se nahradí novým. Všechny změny se projeví až po opětovném přihlášení.

Pro potřeby e-shopu založíme profil zaměstnanec, kde bude využit limit parametrů hesla, takže se specifikuje, kolikrát se bude uživatel moci přihlásit, kolikrát se může chybně přihlásit a za jak dlouho se odemkne zamčený účet.

Příkaz, který vytvoří profil zaměstnanec bude vypadat následovně:

```
CREATE PROFILE employee LIMIT
FAILED_LOGIN_ATTEMPTS 5
PASSWORD_LIFE_TIME 60
PASSWORD_REUSE_TIME 40
PASSWORD_REUSE_MAX UNLIMITED
PASSWORD_VERIFY_FUNCTION verify_function
PASSWORD_LOCK_TIME 3
PASSWORD_GRACE_TIME 10
```

Byl vytvořen profil employee, který specifikuje, že pokud uživatel zadá pětkrát špatné přihlašovací údaje, účet se automaticky zablokuje. Odblokuje se opět po třech hodinách. Heslo je platné po 60 dnů a následujících 10 dnů po ukončení platnosti ho bude ještě možné změnit. Stejné heslo nebude moci použít po dobu 40 dnů po změně. Heslo je kontrolováno funkcí verify_function.

Pokud se vytváří nový uživatel, může se automaticky přiřadit i profil uživatele, a to následujícím způsobem:

```
CREATE USER Lenon IDENTIFIED BY A32ds52aSDQ32a PROFILE
employee;
```

Následně může být vytvořen i profil manažer.

```
CREATE PROFILE shop_manager LIMIT
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LIFE_TIME 40
PASSWORD_REUSE_TIME 50
PASSWORD_REUSE_MAX UNLIMITED
PASSWORD_VERIFY_FUNCTION verify_function
PASSWORD_LOCK_TIME 1
PASSWORD_GRACE_TIME 15
```

Tento příkaz vytvořil profil manažera obchodu. Po 3 neúspěšných pokusech o přihlášení se účet zablokuje a heslo se znovu odblokuje po jedné hodině. Heslo je platné 40

dnů a během dalších 15 dnů jej bude možné změnit. Stejné heslo může být znovu použito po 50 dnech. Stejně jako u zaměstnance, i zde je heslo kontrolováno funkcí verify_function.

V tomto duchu se může vytvořit nejen uživatel přistupující k e-shopu zevnitř, ale také zvenku. Vytvoří se profil zákazníka a přesně u tohoto profilu lze také krásně využít password_parameters. Pokud se zákazník vícekrát přihlásí do e-shopu špatnými přihlašovacími údaji, jeho účet se zablokuje a zákazník si bude moci požádat o dočasné heslo, které mu přijde na e-mail. Dočasné heslo bude platit jen určitou dobu a zákazník si ho bude muset změnit v požadovaném rozsahu a složitosti. V tomto případě je důležité, aby se heslo, které zákazník uvedl při registraci korespondovalo s jeho adresou, jinak heslo nebude doručeno a účet se nebude moci obnovit. Samozřejmě je tu možnost kontaktovat přímo podporu e-shopu a určitě by byla nalezena cesta, jak tento případ vyřešit.

5 Ověření a vyhodnocení

Naše vyhodnocení probíhalo následovně, vytvořili se účty jednotlivých uživatelů a byla na nich testována funkčnost přihlašování, změna hesla při zapomenutí a následně i přihlášení s heslem nesprávným. Výsledkem nám byla 100% úspěšnost konečného přihlášení se do systému. V případě, kdy uživatel zadal nesprávné heslo systém jej vyzval k opakování akce a v konkrétním případě testování profilu shop_manager se profil po 3 neúspěšných pokusech choval dle nastavení (například, hodinové uzamčení vstupu).

5.1 Posouzení integrity dat

Od počátku, kdy data vkládáme do počítače, až do té chvíle, kdy se k nim opravdu dostaneme, projdou několika procesy, při kterých nastávají určitá rizika. Je zde úskalí v možné manipulaci, ať už ze strany lidí nebo samotného informačního systému. Běžně se tedy stává, že vztahy mezi tabulkami nebo buňkami jsou nepřehledné nebo dochází k jejich záměně. To je hlavní důvod, proč je nezbytný test integrity.

5.2 Srovnání s komerčními databázemi

V této kapitole shrnu srovnání databáze, která byla vytvořena na míru e-shopu a databáze, která je vytvořena automaticky s instalací open source systému, jako je například WordPress. Na základě zkušeností z praxe zde vytknu pár věcí, které jsou dle mého názoru nejvíce podstatné.

5.2.1 Nevýhody

Největším úskalím je zde čas, tedy při vyšším počtu dat se rychlost databáze výrazně sníží. Komerční databáze jsou také snadněji napadnutelné, tedy náchylnější pro nežádoucí vstupy, jak jsem již zmiňoval. Mnohdy také nastává problém v komunikaci s databází a ztrácíme v ní přehled. To je zapříčiněno i tím, že jednotlivé pluginy si často vytváří vlastní tabulky. S pluginy se váže spoustu dalších nevýhod a to například, že plugin pokaždé vytváří jiný člověk, a tak zde nastává riziko nesprávného fungování tabulek po jeho aktualizaci. S každou další instalací a aktualizací pluginu vystavujeme databázi možnému výpadku, tedy že si můžeme databázi shodit. Vždy není zcela jasné, co která tabulka

uchovává a při objemné šabloně a velkém počtu pluginu se databáze stává těžko udržovatelnou. Jednou z hlavních příčin jsou pak redundantní data, tedy že se v databázi věci doublují neboli vyskytují víckrát a zcela zbytečně, například metadata. Není zabezpečená integrita dat a stává se, že po vymazání určitého záznamu, se odstraní i data navázaná na tento záznam.

5.2.2 **Výhody**

Stěžejní výhodou je zde snadné vytvoření. Člověk nemusí být z oboru, aby tuto činnost zvládl. Stačí zadat pouze server, kde je databáze vytvořena, následně údaje pro přihlášení do databáze (host, username a heslo a název databáze) a systém v tuto chvíli databázi automaticky vytvoří. Komerční databáze je většinou zdarma, což je často hlavním klíčem v rozhodování uživatelů. Člověk si nemusí hledat a platit odborníka a vše mu automaticky generuje systém. I v tomto případě je ale možné si po čase část databáze dotvořit individuálně, tedy dodělat si tabulky dle vlastní potřeby. K tomu však potřebujeme osobu, která je znalá v programování a návrhu databáze.

6 Závěr

Při začátku této práce byl stanovený cíl, a to takový, vytvořit relační databázi fiktivního internetového obchodu, která bude zabezpečená na úrovní databáze. Jelikož tato práce neřeší celý záběr e-shopu, ale řeší základní část, ve které jsou specifikovány základní principy fungování, byly na začátku také vytknuta omezení této práce. Struktura databáze obsahuje reálné prvky, které se běžně vyskytují v internetových obchodech a lze na ní tedy modelovat základní vazby mezi entitami a docílit tak zabezpečení databáze. V teoretické části byly použity informace, ze kterých se následně celá část skládala.

V dnešním světě, kdy jsou internetové obchody stále běžnější a je čím dál jednodušší si je vytvořit, jelikož je spousta redakčních systémů, které nabízejí e-shopové řešení (např. Wordpresss, Shoptet), ale na druhou stranu, ne každý řeší zabezpečení své databáze nebo o nějaké databázi nemá mnohdy ani ponětí. Zde je ten bod, kdy se mohou odlišit e-shopy, které byly vytvořeny klikáním šablony a ty, nad nimiž pracoval tým lidí, kteří řešili jednotlivé komponenty separátně. Je tu v podstatě několik důvodů, proč mít dobrou a zabezpečenou databázi a mezi ty hlavní patří určitě rychlost, smysluplnost obchodu, a hlavně v dnešní době tolik citlivé uchovávání osobních údajů.

Dobře navržená a normalizovaná databáze nebude mít problém se zpracováním velkého objemu dat a bude stabilní, což se o databázích, které poskytují již zmiňované redakční systémy, říct nemůže. Z databáze by měl být patrný i jakýsi model obchodu, jeho fungování a vnitřní infrastruktura, takže je taky důležité, aby byly jednotlivé tabulky pojmenovány přesně podle toho, jaká je jejich funkce. Toto je také mnohdy opomíjené.

Je tedy vytvořená modelová společnost se svým internetovým obchodem, je navrhnutá a normalizovaná databáze a přechází ta hlavní a velmi důležitá část, a to je zabezpečení.

Zabezpečení databáze může být na několika úrovních, záleží také na tom, jaké data jsou v databázi uchovávány a kdo k jakým datům má přístup, z čehož vyplívá, že musí být také jasně definované pozice a k těmto pozicím přiřazeni zaměstnanci. Pokud je jasně vymezeno, kdo bude správce databáze a kdo bude mít přístup do určitých částí struktury, databáze se otestuje. Otestují se jednotlivý uživatelé, role, vyzkouší se pohledy. Pokud je vše v pořádku, databáze se může začít užívat.

Pro každou společnost by ochrana dat svých zákazníků a dodavatelů měla být na prvním místě. Pokud data z databáze uniknou, nebo se ztratí vede to pak k problémům jak se zákazníky, jelikož o ně přijde, tak i s dodavateli. Jelikož tyto data, podle občanského zákoníku, podléhají utajení, stejně jako ve vnitropodnikové politice, mohla by se firma snadno dostat do právních sporů.

Veškerá data, která byla v databázi vytvořena byla zabezpečena na několika úrovních. Od základního omezení heslem, bez kterého se nelze přihlásit do databáze a nelze pracovat ani prohlížet její data, přes uživatele, kteří byli vytvořeni v rámci vnitropodnikové hierarchie. Role, které jasně vymezili, který uživatel může dělat úpravy v databázi a který může jen prohlížet data a také byly tvořeny na základě hierarchie firmy. Často využívanou metodou byly pohledy, který nám umožní zobrazit pouze náhled části dat bez možnosti samotné úpravy dat. Profily, které jsou vytvořeny pro externí zabezpečení rozšiřují podstatu zabezpečení uživatelských účtů.

Čtenáři této bakalářské práce by mělo být jasné jaké výhody nese databáze postavená na míru konkrétním potřebám a s týmem odborníků oproti té, kterou si nakliká pomocí již přednastavených šablon. Zároveň, pokud ho všechna tato fakta opravdu dovedou k rozhodnutí sestavení unikátní databáze, je zde v rámci praktické části nastíněna tvorba, Tedy všechny základy s tím spojené, co musí databáze zahrnovat, na co je dobré si dát pozor a čemu se striktně vyvarovat. Benefitem je pak celkové schéma, které slouží pro představu člověku, který jako neznalý v oboru, poptává tým odborníků k navrhnutí databáze pro potřeby jeho e-shopu.

Závěrem jsme si srovnali komerční databázi s tou, která je dělána na míru a ukázali si pozitiva i úskalí té komerční. Jako největší pozitivum byla dostupnost a jednoduchost, ale naopak úskalí je zde v rychlosti, závadnosti a špatné údržbě. Čtenář si tak může udělat resumé jaký směr ukládání dat si vybere.

Toto téma je stále aktuální a určitě tomu i tak bude, jelikož se kvůli koronavirové krizi přesouvá spousta firem do oblasti on-linu, do kterého spadají právě e-shopy.

Probíraná oblast nám přináší stále nové poznatky, jak je s databází pracováno a jak ji ještě lépe zabezpečit proti útokům.

7 Seznam použitých zdrojů

7.1 Knižní

1. URMAN, Scott, Ron HARDMAN a Michael MCLAUGHLIN. Oracle: programování v PL/SQL. Brno: Computer Press, 2008. Programování (Computer Press). ISBN 978-80-251-1870-2.

2. MORKES, David. Microsoft SQL Server 2000: tvorba, úprava a správa databází. Praha: Grada, 2004. Podrobný průvodce začínajícího uživatele. ISBN 8024707322.

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

4. WALTERS, R. E. Mistrovství v Microsoft SQL Server 2008: [kompletní průvodce databázového experta]. Brno: Computer Press, 2009. ISBN 978-80-251-2329-4.

5. BÍLA, Jiří a František KRÁL. Databázové a znalostní systémy. Praha: České vysoké učení technické, 1999. ISBN 80-01-01925-X.

6. GROFF, James R. a Paul N. WEINBERG. SQL: kompletní průvodce. Brno: CP Books, 2005. Programování. ISBN 80-251-0369-2.

7. POKORNÝ, Jaroslav. Počítačové databáze. Praha: Kancelářské stroje, 1991. Výběr informací z organizační a výpočetní techniky.

8. KNIGHT, Brian. Microsoft SQL Server 2000: pokročilé techniky. Brno: Computer Press, 2004. ISBN 80-251-0111-8.

7.2 Internetové

9. Kompletní e-shop v Nette. ITnetwork.cz [online]. Praha, [2019], 2016 [cit. 2019-08-27]. Dostupné z: https://www.itnetwork.cz/php/nette/e-shop.

10. Oracle CREATE PROFILE. Oracle Tutorial [online]. USA: Oracle Tutorial, Copyright © 2021 [cit. 2021-02-24]. Dostupné z: https://www.oracletutorial.com/oracle-administration/oracle-create-profile/

11. CREATE PROFILE. Documentation [online]. USA: Oracle and/or its affiliates, Copyright © 2021 [cit. 2021-02-24]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6010.htm

12. Technika útoku SQL injection. ITnetwork.cz [online]. Praha: CID:IGN-C276046-JHR David Čápka, 2014 [cit. 2021-03-15]. Dostupné z: https://www.itnetwork.cz/php/bezpecnost/technika-utoku-sql-injection

7.3 Obrázky

13. Alza.cz [online]. Praha: Aleš Zavoral, c1994 - 2021 [cit. 2021-02-11]. Dostupné z: https://www.alza.cz/

14. Essenté [online]. Praze: Webmium, 2017 [cit. 2021-02-11]. Dostupné z: https://eshop.essente.cz/