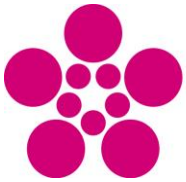


Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Diplomová práce

Tvorba datového modelu vybrané agendy ekonomického informačního systému podniku

Vypracoval: Bc. Josef Soukup

Vedoucí práce: Ing. Petr Hanzal, Ph.D.

České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Josef SOUKUP**
Osobní číslo:	**E21040**
Studijní program:	**N0613A140025 Aplikovaná informatika**
Specializace:	**Podniková informatika**
Téma práce:	**Tvorba datového modelu vybrané agendy ekonomického informačního systému podniku**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je analýza vybrané agendy podniku, tvorba konceptuálního a fyzického datového modelu informačního systému pro tuto agendu. Praktická část se bude zabývat realizací tohoto datového modelu na vybraném databázovém systému.

Metodický postup:

1. Studium odborné literatury a problematiky, vyhledání relevantních zdrojů související s problematikou.
2. Analýza požadavků vybrané agendy.
3. Výběr vhodného systému pro tvorbu datového modelu.
4. Tvorba vlastního konceptuálního a fyzického datového modelu.
5. Zhodnocení vytvořeného modelu.
6. Závěr.

Rozsah pracovní zprávy:	**50 – 60 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

1. BUCHALCEVOVÁ, A. (2009). *Metodiky budování informačních systémů.* Praha: Oeconomica.
2. GÁLA, L., POUR, J., & ŠEDIVÁ, Z. (2015). *Podniková informatika: Počítačové aplikace o podnikové a mezipodnikové praxi.* Praha: Grada Publishing.
3. BEYNON-DAVIES, P. (2017). *Database systems.* Bloomsbury Publishing.
4. LI, Qi, & CHEN, Y. (2009). Entity-relationship diagram. In: *Modeling and Analysis of Enterprise and Information Systems.* Springer, Berlin, Heidelberg, s. 125-139.
5. SILBERSCHATZ, A. & at all. (1991). Database systems: Achievements and opportunities. *Communications of the ACM.* 34.10, s. 110-120.

Vedoucí diplomové práce: **Ing. Petr Hanzal, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: **11. ledna 2022**
Termín odevzdání diplomové práce: **14. dubna 2023**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 9. března 2022

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 14. 4. 2023

Podpis studenta

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Petru Hanzalovi, Ph.D. za odborný přístup k vedení mé práce a jeho užitečné rady. Poděkování patří také mé partnerce Bc. Michaele Doležalové za cenné rady, objektivní kritiku a poskytnutí zpětné vazby.

Obsah

1 Úvod

Informační systémy jsou pro malé a střední podniky klíčové pro zvyšování efektivity, zvyšování konkurenceschopnosti a udržení kroku s globální konkurencí. Tyto systémy jim umožňují automatizovat procesy, snížit chyby a zefektivnit komunikaci mezi jednotlivými odděleními a se zákazníky. Proto je jejich poptávka po nich v poslední době stále vyšší. Tyto podniky čelí stejným výzvám jako velké společnosti, jako jsou například potřeba efektivní správy dat, automatizace procesů a zvyšování produktivity, ale často mají menší rozpočet a méně zdrojů na implementaci informačních systémů.

Ekonomické agendy podniků mohou obsahovat plány aktivit a opatření, které jsou zaměřeny na dosahování ekonomický cílů podniku. Tyto cíle mohou být například zaměřené na zisk, růst, zlepšení konkurenceschopnosti nebo na zvýšení tržního podílu.

Hlavním cílem této práce je analýza agendy ekonomického informačního systému malé kamenické firmy v Jihočeském kraji a aplikace postupů a metod při tvorbě datového modelu této agendy. Dílčím cílem je ověření funkčnosti datového modelu na vybrané platformě a konzultace využitelnosti s obchodním manažerem této firmy.

V teoretické části práce je popis informačních systémů a jejich rozdělení. Procesní analýza a její využitelnost. Dále je zde charakterizována podstata datového modelování a popis jeho vlastností a náležitostí. Jsou zde uvedené programovací prostředí, ve kterých lze datové modelování uskutečnit. Následuje popis datových typů, které jsou využity při modelování. Dále je uvedena charakteristika skriptovacího jazyka SQL s jeho náležitostmi.

Praktická část práce je zaměřena na analýzu prostředí firmy a tvorbu datového modelu informačního systému s využitím dostupných nástrojů a dalších metod a postupů uvedených v teoretické části. Pro tvorbu modelu je vybrán vhodný modelovací nástroj. Vytvořený datový model je otestován na platformě bit.io, kam je nahrán zdrojový kód. A je konzultován s obchodním manažerem firmy.

2 Informační systém

Aby dávalo smysl používat informační systém (IS) je nutné mít nějaká data, se kterými bude pracovat. Data jsou však podle Rowleyové (2007) pouze diskrétní, objektivní fakta nebo pozorování, které z důvodu nedostatku kontextu a interpretace nemají hodnotu, neboť nejsou nijak zpracována ani nijak organizována. Může tak jít o různé signály, stimuly nebo symboly, které nejsou prozatím k užitku. Aby byla data užitečná, je nutné z nich získat nějakou informaci.

Pomocí uspořádání dat podle Rowleyové (2007) lze z dat získat smysl, význam a relevanci. Tímto způsobem lze získat ucelený soubor dat, který je možné interpretovat a lze tak obdržet užitečnou informaci. Informace tak má větší hodnotu než data samotná, protože se ve strukturovaných datech lze lépe orientovat.

Informační systém je pak sada nástrojů, technologií a procesů, které spolu souvisí a jsou navrženy tak, aby podporovaly řízení těchto informací a dalších zdrojů v podniku. Tyto systémy jsou využívány jako centrální platforma pro shromažďování, zpracování, uchovávání a sdílení informací, což ve výsledku pomáhá firmám zvýšit efektivnost a informovanost při rozhodování.

IS se skládá z řady komponentů, jako jsou hardware, software, databáze, sítě a lidské zdroje. Tyto komponenty spolu úzce souvisí a vzájemně se doplňují, aby poskytovaly ucelený a funkční systém.

Jak uvádí Buchalcevová (2009), tak takový systém je navržen pro specifické účely, jako je řízení zákaznických vztahů, řízení dodavatelského řetězce nebo řízení lidských zdrojů. Tento systém může být také navržený pro celkové podnikové řízení, jako je ERP. U IS je kladen důraz na jednoduchost použití, snadnou integraci stávajícího podnikového prostředí a poskytování relevantních a aktuálních informací pro řízení. Správně implementovaný a používaný IS může významně pomoci firmám zlepšit své podnikové procesy, zvýšit efektivitu a optimalizovat využití zdrojů.

2.1 Rozdělení IS

Existuje mnoho druhů podnikových informačních systémů, kde mezi nejběžnější podle Brucknera et al. (2012) patří následující:

ERP (Enterprise Resource Planning) systémy

ERP systém integruje různé části podniku, jako jsou účetnictví, skladování, výroba a prodej. A poskytují centralizovaný přehled o celkových aktivitách podniku.

CRM[1] systémy

CRM systém je využíván ke správě vztahů s klienty a poskytuje informace o historii komunikace, kontaktech a obchodních příležitostech.

SCM[2] systémy

SCM systém se zaměřuje na řízení dodavatelského řetězce a poskytuje informace o stavu objednávek, skladových zásobách a logistických procesech.

BI[3] systémy

BI systém poskytuje podnikům informace o svých datech a umožňují jim vytvářet reporty, analýzy a vizualizace pro lepší rozhodování.

HRIS[4] systémy

HRIS systém se zaměřuje na řízení lidských zdrojů a poskytuje informace o zaměstnancích, plánování rozpočtu a mezd.

2.2 ERP systém

Pro propojení jednotlivých ekonomických úseků uvnitř podniku se dle SAP (2022) nejvíce využívají ERP, které podporují řízení různých podnikových procesů. Tyto systémy integrují informace z různých částí podniku a poskytují centralizovaný přehled o celkových aktivitách podniku.

2.2.1 Náležitosti ERP systémů

Podle SAP (2022) jsou důležitými částmi systému následující prvky:

Společná databáze

Taková databáze poskytuje centralizované informace v jednotné verzi

Integrované analytické nástroje

Jedná se o vestavěné analytické nástroje, samoobslužné nástroje BI, výkaznictví a dodržování

[1] Customer Relationship Management
[2] Supply Chain Management
[3] Business Intelligence
[4] Human Resource Information System

předpisů, které mohou poskytovat inteligentní analýzy pro jakoukoli oblast podnikání.

Vizualizace dat

Tento prvek umožňuje vizuální prezentaci klíčových informací pomocí dashboardů a KPI[5], které pomáhají při potřebě rychlého rozhodování na základě dostupných informací.

Automatizace

Automatizace umožňující opakující se běh úloh a pokročilých RPA[6] s využitím umělé inteligence a strojového učení.

Integrace

Hladká integrace podnikových procesů a workflow, ať už s ostatními softwarovými řešeními a zdroji dat, tak i s těmi od třetích stran.

Nadnárodní podpora

Taková podpora by měla zahrnovat jazyky, měny a místní obchodní postupy a předpisy. Podobně jako technická podpora pro cloudové služby zahrnuje školení, helpdesk a implementaci.

Volba nasazení

Nasadit ERP lze ve třech variantách - Cloud[7], on-premise[8] nebo hybridní.[9]

2.2.2 Výhody ERP

Existuje spousta aspektů, které vyzdvihují ERP systémy, níže jsou však popsány jen ty nejdůležitější podle Li & Zhao (2006).

Vyšší produktivita

Ve smyslu zjednodušení a automatizace hlavních podnikových procesů a poskytnutí pomoci všem v organizaci více pracovat s menším počtem zdrojů.

---

[5] *Key performance indicator* – Klíčové ukazatele výkonnosti
[6] *Robotic process automation* – Robotická automatizace procesů
[7] Cloud - S cloudovým ERP je software hostován v cloudu a dodáván přes internet jako služba. Poskytovatel softwaru obecně dbá na pravidelnou údržbu, aktualizace a zabezpečení jménem klienta. Cloudové ERP je nejoblíbenější metodou nasazení z důvodů nižších počátečních nákladů, větší škálovatelnosti a agility, snazší integrace a dalších.
[8] On-premise - Jedná se o tradiční model pro nasazení softwaru, kde vše ovládá klient. ERP systém je obvykle nainstalován v datovém centru klienta v místech dle jeho výběru. Za instalaci a údržbu hardwaru a softwaru zodpovídá klient.
[9] Hybridní ERP – Některé aplikace ERP jsou provozovány v cloudu a některé u klienta.

Hlubší přehledy

Jedná se o eliminaci informačních sil[10], získání důvěryhodného zdroje dat a možnost získat rychlé odpovědi na důležité podnikové otázky.

Zrychlené vykazování

Jde o rychlé obchodní a finanční výkaznictví a snadné sdílení výsledků, práci na přehledech a zvyšování výkonu v reálném čase.

Nižší riziko

Jedná se o zajištění shody s regulačními požadavky a předvídání či předcházení rizikům.

Jednodušší IT

Použitím integrovaných ERP aplikací, které sdílejí databázi, lze IT zjednodušit a poskytnout všem snadnější způsob práce.

Lepší agilita

Pomocí efektivních operací a připravenému přístupu k datům v reálném čase lze rychle identifikovat nové příležitosti a reagovat na ně.

2.2.3 Nevýhody ERP

Vysoké náklady

Implementace a údržba ERP systému mohou být nákladné, zejména pro menší společnosti.

Komplexnost

ERP systémy mohou být složité na implementaci a používání, což může vést k potížím s trénováním zaměstnanců a zajištěním správného fungování.

Nepřizpůsobitelnost

Některé ERP systémy mohou být těžko přizpůsobitelné specifickým požadavkům podniku, což může vést k neuspokojivému výkonu.

Ztráta dat

Při přechodu na ERP systém mohou být některá data ztracena nebo přepsána, což může mít

[10] Informační silo je systém správy informací, který není schopen svobodně komunikovat s jinými systémy správy informací. Komunikace v informačním sila je vždy vertikální, což ztěžuje nebo znemožňuje systému pracovat s nesouvisejícími systémy.

negativní dopad na podnikové procesy.

Závislost na dodavateli

Podniky mohou být závislé na dodavateli ERP systému pro podporu a údržbu, což může být problematické, pokud dodavatel neposkytuje adekvátní podporu nebo se stane nedostupným.

Změna zvyklostí

Implementace ERP systému může vést ke změnám ve zvyklostech zaměstnanců a může být obtížné přizpůsobit se těmto změnám.

2.3 Logistický ERP systém

Pro integraci procesů logistiky s dalšími oblastmi podniku, jako jsou finance, výroba, prodej a další je určen ERP systém zaměřený na logistiku Tento systém pomáhá firmám plánovat, řídit a sledovat pohyb zboží a materiálů, což zlepšuje efektivitu a produktivitu výroby a distribuce.

Důležitou funkcí tohoto systému je správa skladu a materiálového hospodářství. Systém umožňuje sledovat stav skladových zásob a plánovat nákupy v souladu s poptávkou. Tento proces je důležitý pro minimalizaci ztrát způsobených nadbytečnými zásobami a zároveň zajistí, že je dostatek materiálů pro výrobu a prodej.

Další důležitou funkcí takového systému je sledování a řízení dodávek. Systém umožňuje plánovat a koordinovat dodávky, což vede k lepšímu řízení zásob a minimalizaci zpoždění v distribuci. Díky tomu může firma lépe reagovat na poptávku zákazníků a zlepšit celkovou spokojenost zákazníků.

Podstatnou funkcí je také plánování a řízení výroby. Systém umožňuje sledovat výrobní procesy a zdroje, aby bylo možné plánovat výrobu efektivněji a minimalizovat zpoždění a zbytečné náklady. Systém také umožňuje sledovat výrobní procesy v reálném čase, což umožňuje rychlé a účinné řešení problémů, které se mohou vyskytnout.

Kromě těchto funkcí existuje mnoho dalších funkcí, jako jsou řízení kvality, sledování výrobních nákladů, plánování dopravy a řízení vztahů se zákazníky. Všechny tyto funkce společně umožňují firmám lépe plánovat, řídit a sledovat procesy v celém obchodním prostředí.

Výhodou takového systému je centralizace informací. Systém umožňuje centralizované uložení informací o výrobě nebo skladových zásobách a umožňuje snadné a rychlé

vyhledávání dat a zároveň usnadňuje komunikaci a spolupráci mezi různými odděleními a pracovníky v podniku.

Jak ve své studii uvádí Muscatello et al. (2003), tak implementace takového systému může být náročná a drahá, zejména pro menší podniky. Vyžaduje úpravy firemních procesů a návyky zaměstnanců, a to pro ně může být obtížné.

2.4 Procesní analýza pro IS

Procesní analýza je klíčovým krokem při návrhu a implementaci informačního systému. Spočívá v podrobném zkoumání a mapování existujících podnikových procesů, aby bylo možné lépe porozumět, jak procesy fungují a jak mohou být vylepšeny pomocí IS.

Tato analýza umožňuje podle Dumase a kol. (2018) zjistit, jaké jsou hlavní kroky v procesu, kdo je zodpovědný za každý krok, jaké jsou vstupy a výstupy procesu, jaký je časový plán a jaké jsou potřebné zdroje. Aalst (2016) dále rozvádí, že tato znalost je důležitá pro tvorbu IS, protože pomáhá analytikům a vývojářům porozumět, jak může IS usnadnit a zefektivnit podnikové procesy.

Procesní analýza pomáhá také odhalit neefektivní kroky v procesu a předložit návrhy na zlepšení. IS může být navržen tak, aby tyto neefektivní kroky odstranil nebo minimalizoval a aby se proces stal rychlejší, levnější a přesnější. Výstupem procesní analýzy jsou procesní diagramy.

2.4.1 Vizualizace procesů

K vizualizace procesů a identifikaci oblastí pro použití IS ke zlepšení efektivnosti lze využít standardizovanou notaci BPMN[11].

Tato notace se skládá ze symbolů, které reprezentují různé typy aktivit, toků dat, událostí a bran. Každý symbol má svůj význam a umožňuje popsat konkrétní aspekt procesu. Např. symbol kruhu reprezentuje událost, symbol šipky znamená tok dat a symbol brány reprezentuje rozhodovací bod v procesu.

Podle Mierse & Whitea (2008) notace dále umožňuje popsat procesy včetně paralelních toků, větvících se cest a dalších aspektů. Umožňuje i propojení procesů s dalšími procesy v rámci

[11] Business Process Model and Notation (BPMN) je grafická notace (soubor grafických objektů a pravidel, podle nichž jsou mezi sebou objekty spojovány) sloužící k modelování podnikových procesů pomocí procesních diagramů.

organizace a umožňuje také popsat provázanost s dalšími systémy. Je využívána pro modelování procesů v oblasti managementu procesů, řízení projektů, automatizace procesů, analýzy procesů a dalších.

Příkladem využití BPMN může být např. vývoj nového produktu. Proces vývoje nového produktu je složitý a zahrnuje mnoho aktivit. Notace BPMN umožňuje snadno vizualizovat tento proces a identifikovat klíčové kroky, na kterých může organizace zlepšit tento proces. V něm může být klíčovým krokem testování produktu. BPMN umožní vizualizovat tento krok v procesu a zjistit, zda jsou v tomto kroku nějaké zbytečné aktivity, které mohou být eliminovány.

2.4.2 Prvky notace BPMN

BPMN používá různé prvky nebo objekty pro vizualizaci a popis podnikových procesů. Následující prvky nebo objekty jsou nejpoužívanější a základní prvky BPMN:

Události

Události reprezentují všechny významné body v procesu, kdy se něco stane. Události mohou být způsobeny interními nebo externími faktory. Existují tři hlavní typy událostí: začátek, střed a konec procesu. Události jsou znázorněny kruhovým symbolem.

Aktivity

Aktivity jsou kroky v procesu, které musí být dokončeny, aby se proces posunul dál. Aktivity mohou být automatické, když se proces automaticky provádí, nebo manuální, když musí být aktivita provedena člověkem. Aktivity jsou znázorněny obdélníkovým symbolem.

Toky procesu

Toky procesu reprezentují spojení mezi různými událostmi a aktivitami v procesu. Existují dva hlavní typy toků procesu: sekvenční a paralelní. Sekvenční toky procesu znamenají, že jedna událost nebo aktivita musí být dokončena, aby se mohla začít další. Paralelní toky procesu znamenají, že několik událostí nebo aktivit může být prováděno současně. Toky procesu jsou znázorněny šipkou.

Brány

Brány se používají v BPMN pro specifikaci větvících se cest nebo podmínek, které musí být splněny, aby proces mohl pokračovat dál. Existují tři hlavní typy bran: exkluzivní (XOR), inkluzivní (OR) a paralelní (AND). Brány jsou znázorněny diamantovým symbolem.

3 Datové modelování

Jednou ze stěžejních součástí informačního systému je dle Beynon-Daviese (2003) datový model, který určuje, jak budou data v systému organizována, jakým způsobem budou vzájemně propojena a jakým způsobem budou přístupná aplikacím a uživatelům. Jinými slovy zachycuje část reality, která má být v IS zobrazena. Datový model ovlivňuje rychlost a spolehlivost systému a určuje, jaké informace budou shromažďovány a analyzovány.

3.1 Konceptuální úroveň modelu

Konceptuální úroveň datového modelování je podle Howea (2001) nejvyšší úroveň abstrakce v procesu datového modelování. Je to místo, kde se popisuje celková logika a struktura dat bez konkrétního zaměření na technické detaily implementace. Cílem konceptuálního modelu je jasně a srozumitelně popsat a reprezentovat podstatu datového problému, aby bylo možné snadno a jednoznačně pochopit a komunikovat návrh datového řešení. Tato úroveň je důležitá, protože umožňuje jednotný pohled na datový problém a napomáhá včasnému odhalení a řešení potenciálních problémů s daty.

V konceptuálním modelu jsou definovány entity, atributy a vztahy mezi nimi, ale nepopisují se žádné konkrétní technické detaily jako datový typ nebo velikost pole. Tyto detaily jsou řešeny na dalších úrovních modelování viz logická a fyzická úroveň v dalších kapitolách, pro které je tato úroveň základem.

3.1.1 Entita

Pojem entita je používán pro označení objektu v databázovém systému, který reprezentuje reálnou věc, jako např. zákazníka, produkt, objednávku apod. Entita má jednoznačný identifikátor a atributy popisující její vlastnosti. Jak uvádí Chlapek a kol. (2019), tak následná práce s entitami probíhá pomocí instancí těchto entit. Pod tím si lze představit, že každý záznam tabulky představuje jednu instanci dané entity.

3.1.2 Atribut

Atribut je vlastnost nebo informace o entitě. Pomáhá identifikovat a charakterizovat entitu a uchovává specifické informace o ní. Mohou nabývat datových typů jako číslo, text, datum, logický typ atd. Každý atribut má název a definici, která určuje, co informace uchovává. Atribut může být také označený jako primární klíč, který jednoznačně identifikuje entitu, nebo jako cizí klíč, který ukazuje na jinou entitu viz kapitola 3.1.3.

Výběr správných atributů pro každou entitu a specifikace jejich datových typů a omezení jsou klíčovými fázemi datového modelování. Tyto informace umožňují přesné a úplné reprezentování informací v datovém modelu a ovlivňují následnou implementaci databáze.

K atributu se může vztahovat doména, která je využívána k definování rozsahu hodnot, které může daný atribut obsahovat v databázi. Zároveň pomáhá při validaci dat, jelikož ukládání hodnot, které nejsou v rozsahu dané datové domény, není možné. Příkladem domény je "Emailová adresa", která by mohla být definována jako "textový řetězec o délce až 255 znaků, který obsahuje symbol @ a má formát něco@něco.cz".

3.1.3 Klíče

Klíč je využíván k odlišení jednotlivých záznamů a zajištění integrity dat. Howea (2001) uvádí, že klíč je v datovém modelování unikátní identifikátor, který se používá k určení jedinečnosti řádku v databázové tabulce.

Podle něj existují následující druhy klíčů v datovém modelování:

Primární klíč

Primární klíč je klíč, který je použit jako hlavní identifikátor řádku v tabulce. Může být složený z jednoho nebo více atributů a musí být unikátní pro každý řádek v tabulce.

Kandidátní klíč

Kandidátní klíč je alternativní klíč, který může být použit jako primární klíč. Tyto klíče musí být také unikátní a neměnné.

Cizí klíč

Cizí klíč je klíč, který odkazuje na primární klíč v jiné tabulce. Tento klíč je využíván k navázání vztahu mezi tabulkami.

Kompozitní klíč

Kompozitní klíč je složený z více atributů a je využíván jako primární klíč pro řádek v tabulce. Tento klíč je vytvořen, pokud žádný atribut samostatně není dostatečně unikátní pro určení jedinečnosti řádku.

3.1.4 Vztahy

Vztahy nebo relace v datovém modelování představují logické spojení mezi entitami. Tyto

vztahy definují, jak jsou entity propojeny a jakým způsobem se informace mezi nimi přenášejí. Relace pomáhají určovat vzájemné souvislosti mezi entitami a umožňují o nich efektivně vyhledávat a ukládat informace. Výběr správného typu relace pro každý případ je klíčovým faktorem pro správné nastavení databáze. Každá relace je specifikována pomocí grafického znázornění.

V datovém modelování existují následující typy relací:

Mnoho k jednomu (1:N)

Obecně je tato relace nejvyužívanější. Lze demonstrovat na příkladu databáze pro sledování objednávek, která obsahuje tabulky Zákazník a Objednávka. Zákazník může vytvořit libovolný počet objednávek. To znamená, že pro každého zákazníka uvedeného v tabulce Zákazník může existovat celá řada objednávek zaznamenaných v tabulce Objednávka.

Mnoho k mnoha (M:N)

Pro demonstraci této relace lze využít tabulky Výrobek a Objednávka. Kde jedna objednávka může obsahovat více výrobků. A opačně se jeden výrobek může objevit v mnoha objednávkách. Z tohoto důvodu může pro každý záznam v tabulce Objednávka existovat mnoho záznamů v tabulce Výrobek. Navíc pro každý záznam v tabulce Výrobek může existovat celá řada záznamů v tabulce Objednávka.

Pro vyjádření relace typu M:N, je nutné vytvořit ještě třetí tabulku, která se často nazývá spojená tabulka, která rozdělí relaci typu M:N na dvě relace typu 1:N. Primární klíč z těchto dvou tabulek vložíte do třetí tabulky. Výsledkem je, že třetí tabulka zaznamená každý výskyt nebo instanci relace. Pro tabulky Objednávka a Výrobek v relaci M:N, tak musí být tato relace definována vytvořením dvou relací 1:N s tabulkou pojmenovanou např. Rozpis objednávek. V každé objednávce pak může být uvedeno více výrobků a každý výrobek může být uveden ve více objednávkách.

Jedno k jednomu (1:1)

Tuto relaci lze vystihnout na příkladu Člověk a Občanský průkaz, kde právě jeden průkaz náleží jednomu člověku a jeden člověk může vlastnit právě jeden občanský průkaz.

Tato relace je obvykle využívána k rozdělení rozsáhlejších tabulek, protože obě takto spojené tabulky musí obsahovat společné pole.

3.1.5 Referenční integrita

K dalším důležitým vlastnostem relačních databází je referenční integrita, která zajišťuje konzistenci dat v databázi. Jedná se o mechanismus, který zabezpečuje správné a konzistentní propojení dat mezi různými tabulkami v databázi. V praxi to znamená, že referenční integrita definuje omezení a pravidla pro vazby mezi tabulkami. Nejčastější formou omezení je použití cizích klíčů, které vymezují vztah mezi tabulkami, a zajišťují, že hodnoty v cizím klíči odkazující tabulky odpovídají hodnotě primárního klíče cílové tabulky, jak uvádí Simsion & Witt, (2004) ve své knize.. To znamená, že pokud je například v jedné tabulce změněna hodnota primárního klíče, automaticky se změní i hodnota cizího klíče v odkazující tabulce, což zajišťuje konzistenci dat.

3.1.6 ER model

ER[12] model je konceptuální model, který je využíván k popisu datového schématu a vztahů mezi daty. ER model je používán pro navrhování relačních databází a umožňuje definovat entity, atributy a vztahy mezi nimi.

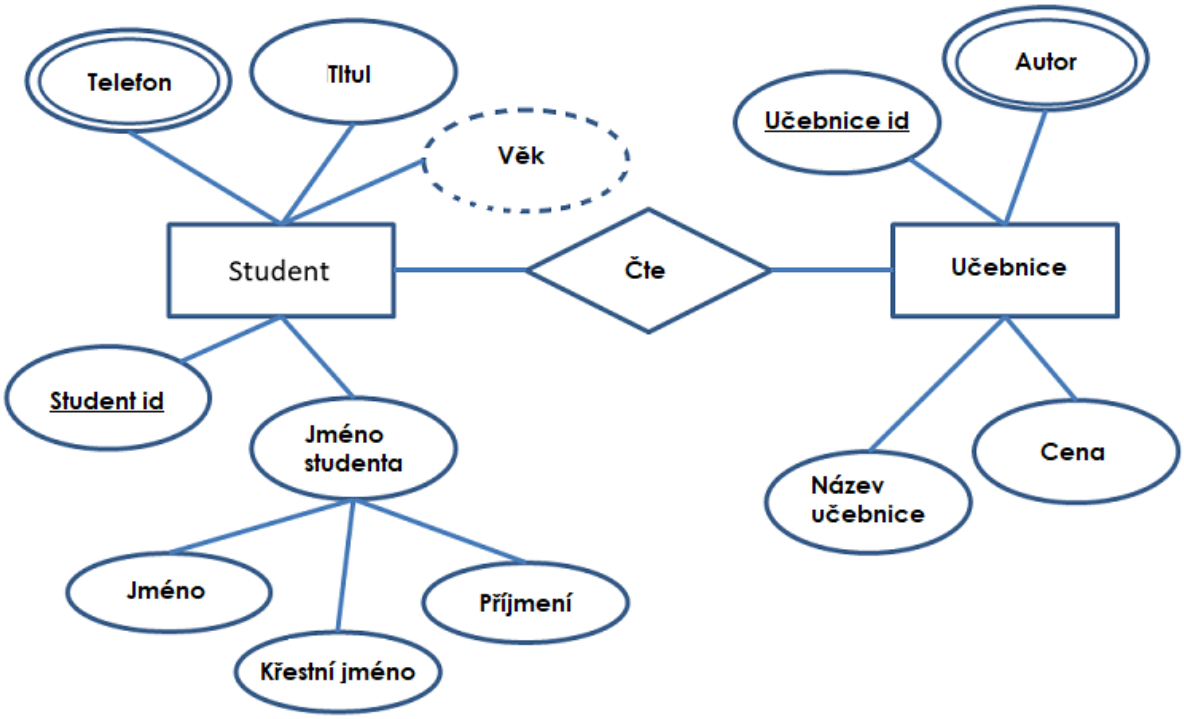
Tento model je založen na grafickém zobrazení entit a vztahů, což umožňuje snadnou komunikaci mezi databázovými specialisty a uživateli. Q. Li & Chen (2009) v ER modelu pro zobrazení entit využívají kruhy nebo obdélníky a vztahy mezi nimi zobrazují jako čáry nebo šipky. Atributy entity se pak zobrazují jako ovály uvnitř entity.

Na následujícím obrázku 1 je znázorněn příklad ER modelu, kde jsou entity Student a Kniha. Tyto entity jsou propojené vztahem Čte. Každá z entit má definované atributy v oválech, kde podtržený atribut značí primární klíč. Dvojitý ovál značí vícehodnotový atribut, avšak takový atribut není optimální viz kapitola 3.4.

Další dělení vztahů přibližuje obrázek 2, který znázorňuje notaci tzv „crow's feet" neboli vraní nohy. Jde o možnost grafického vyjádření vztahů uvedených v kapitole 3.1.4.

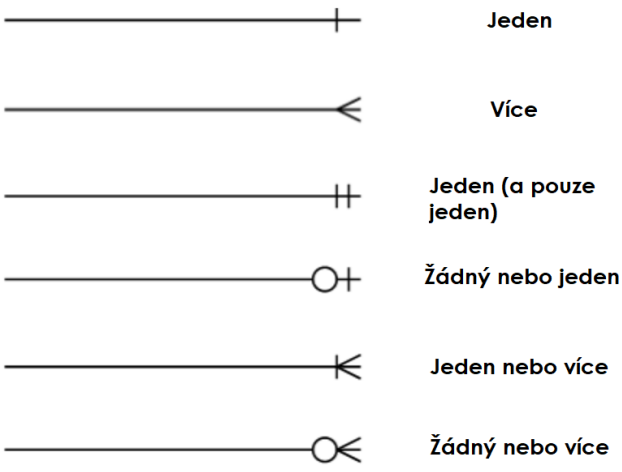
[12] Entity-Relationship

Obrázek 1: Příklad ER modelu



Zdroj: QoChuk (2021)

Obrázek 2: Grafické znázornění vraních nohou



Zdroj: QoChuk (2021)

3.2 Logická úroveň modelu

Jak Howe (2001) dále uvádí, tak logická úroveň datového modelování je střední úroveň abstrakce v procesu datového modelování. Logický model navazuje na konceptuální model a přidává technické detaily implementace, ale stále nezahrnuje konkrétní technické detaily

jako například hardware nebo software.

Cílem logického modelu je specifikovat, jak budou data uložena a jakým způsobem budou data mezi různými entity navzájem propojena. Logický model také určuje datový typ a délku každého atributu a specifikuje, zda má být atribut povinný nebo volitelný.

Logický model je zdrojem pro tvorbu fyzického modelu, který popisuje konkrétní technickou implementaci datového řešení, jako například typ a velikost použitých databázových tabulek a sloupců.

Logická úroveň modelování je důležitá, protože umožňuje ověřit a upravit konceptuální model tak, aby odpovídal skutečným potřebám a požadavkům na datové řešení. Zároveň poskytuje jasný a srozumitelný obraz o tom, jak budou data uložena a jakým způsobem budou propojena, což umožňuje včasné odhalení a řešení potenciálních problémů s daty.

3.2.1 Datové typy

Datové typy v databázových systémech mají různé vlastnosti, které určují, jak se data v těchto typech chovají a jak jsou ukládána. V tabulce 1 níže jsou uvedeny vlastnosti nejčastěji používaných datových typů při datovém modelování.

Tabulka 1: Používané datové typy

Datový typ	Vlastnosti	Využitelnost
Integer	Celá čísla různých délek	Identifikátory, počítadla
Float	Plovoucí desetinná čárka	Čísla s vysokou přesností
Decimal	Pevná desetinná čárka	Ukládání finančních údajů
Varchar	Textové řetězce s proměnnou délkou	Krátké textové informace
Text	Delší textové řetězce než varchar	Delší popisy nebo poznámky
Datetime	Datum a čas	Ukládání datumů událostí
Bit	Hodnota 0 nebo 1	Přepínače, příznaky

Zdroj: vlastní zpracování

3.2.2 Relační databáze a OLAP

Relační databáze je typ databáze, který využívá relačního modelu, aby popsal data a vztahy

mezi daty. Tyto vztahy jsou popsané pomocí relací, které představují matematické množiny sloupců a řádků. Tyto relace se dají kombinovat do větších celků pomocí dotazů, jako je například SELECT, které vrací specifický řetězec dat.

Na druhou stranu, OLAP[13] se vyznačuje schopností efektivně analyzovat a agregovat data v databázi. Tyto databáze bývají navržené tak, aby umožnily rychlé a efektivní dotazy a analýzy. Tyto databáze se často využívají k analýze tržeb, nákladů a dalších podnikových dat.

OLAP je databázový systém, který je využíván k analytickému zpracování dat. Tyto systémy jsou optimalizovány pro rychlé výpočty a sestavování výkazů, a umožňují uživatelům analyzovat data z různých úhlů pohledu. Tyto systémy také umožňují uživatelům provádět složité analýzy dat pomocí nástrojů pro výpočty, jako jsou například KPI[14] nebo sestavování výkazů.

Hlavním rozdílem mezi relačními databázemi a OLAP je způsob, jakým data ukládají a jak jsou přístupná pro analýzu. Relační databáze jsou navržené tak, aby umožňovaly jednoduchý a efektivní přístup ke specifickým datům pomocí dotazů, zatímco OLAP databáze umožňují efektivní a rozsáhlou analýzu dat.

3.3 Fyzická úroveň modelu

Fyzická úroveň datového modelování je dle Howea (2001) nejnižší úroveň abstrakce v procesu datového modelování. Cílem fyzického modelu je specifikovat konkrétní technickou implementaci datového řešení, včetně specifikací hardware a software, které budou použity.

Tento model zahrnuje podrobnosti jako typ a velikost databázových tabulek a sloupců, indexy pro rychlejší vyhledávání dat, souborový systém a místo uložení dat. Také specifikuje kapacitu a výkonnost datového řešení, jako je maximální počet uživatelů a výkon při čtení a zápisu dat.

3.4 Normalizace databáze

Normalizace dat v databázi je proces, který se zabývá optimalizací datových struktur tak, aby se minimalizovala duplicitní a nekonzistentní data, zlepšila integrita dat a zvýšila efektivita

[13] On-Line Analytical Processing
[14] Key Performance Indicators

při zpracování dat. Normalizace probíhá prostřednictvím tvorby a analýzy funkčních závislostí mezi atributy jednotlivých entit. Tyto závislosti se použijí k rozdělení dat do více tabulek, tzv. normalizačních stupňů. Tyto stupně se odvíjejí od počtu funkčních závislostí a počtu atributů v tabulce.

3.4.1 1 NF

První normální forma (1NF) je první stupeň normalizace dat, který zajišťuje, že každý atribut v databázi má hodnotu, která je primitivního datového typu (např. celé číslo, řetězec atd.). Tzn., každý atribut by měl být jedinečný a neměl by se opakovat v rámci jedné entity.

Tento stupeň se také stará o to, že každý atribut má jedinečnou hodnotu a že hodnoty atributů nemohou být dále rozděleny na menší části. Toho se dosáhne tím, že se do jedné tabulky ukládají pouze jednotlivé instance jedné entity. Tyto instance se nazývají řádky a každý atribut se nazývá sloupec.

3.4.2 2 NF

Druhá normální forma (2NF) je druhý stupeň normalizace dat, který vyžaduje, aby byly splněny následující dvě kritéria:

- Databáze by měla být v první normální formě (1NF)
- Každý nesouvisející atribut by měl být umístěn v samostatné tabulce, pokud není vazbou závislý na primárním klíči dané entity.

Toto znamená, že každý atribut v databázi by měl být závislý pouze na primárním klíči entity, kterou reprezentuje. Pokud je atribut závislý na jiném atributu, ale není závislý na primárním klíči, měl by být umístěn v samostatné tabulce. To zabraňuje duplicitnímu ukládání dat a umožňuje efektivní správu dat.

Výsledkem je, že databáze bude mít více tabulek, ale každá tabulka bude mít pouze jeden účel a bude obsahovat pouze relevantní data pro danou entitu.

3.4.3 3 NF

Třetí normální forma (3NF) je třetí stupeň normalizace dat, který vyžaduje, aby byly splněny následující tři kritéria:

- Databáze by měla být v druhé normální formě (2NF).

- Každý atribut, který není závislý na primárním klíči, by měl být závislý na celém primárním klíči.
- Neexistuje žádný funkčně závislý atribut na jiném atributu.

To vyjadřuje, že každý atribut v databázi by měl být pouze závislý na primárním klíči dané entity nebo na celém primárním klíči, ale nikoli na jiném atributu.

Databáze následně bude mít ještě více tabulek, ale každá tabulka bude obsahovat pouze relevantní data a bude mít jasně definovaný účel. Tato forma normalizace poskytuje maximální ochranu proti datovým chybám a zároveň umožňuje efektivnější správu dat.

3.4.4 Boyce-Coddova NF (BCNF)

Jak uvádí Tuteja (2015), tak BCNF je vlastnost relačních databázových schémat, která zajišťuje, že každá závislost funkčně závisí pouze na kandidátním klíči tabulky, tzn. na žádné jiné množině atributů, které nejsou klíčem. Z toho plyne, že každá netriviální funkční závislost (tj. závislost, kde jsou různé hodnoty v jednom atributu spojeny s různými hodnotami v jiném atributu) v tabulce je závislostí na celém kandidátním klíči.

3.4.5 4 NF

Čtvrtá normální forma se stará o to, aby všechny vícehodnotové závislosti byly eliminovány. Podle Tuteja (2015) zde vícehodnotová závislost existuje, když množina atributů závisí na stejné množině klíčů a každý atribut této množiny je závislý na stejné podmnožině klíčů.

3.4.6 5 NF

Nejvyšší formou normalizace je pátá normální forma. Ta podle Tuteja (2015) ošetřuje, aby všechny závislosti byly zachovány pouze mezi jednotlivými kandidátními klíči a žádná jiná množina atributů nemohla být funkčně závislá na žádné jiné množině atributů. To znamená, že každý atribut je závislý pouze na kandidátním klíči.

4 Charakteristika SQL

SQL[15] je standardní skriptovací jazyk pro práci s relačními databázemi. Jeho hlavním účelem je umožňovat uživatelům manipulovat s daty v databázi (např. vkládání, úprava a mazání dat), stejně jako vyhledávání a agregaci dat. Tento jazyk podporuje různé typy dat, jako jsou text, čísla a datum, a umožňuje definovat relace mezi entitami. Jde o transakční jazyk, to znamená, že umožňuje provádět sérii operací jako jednu transakci, která buď celá proběhne, nebo se neprovede vůbec. Jedná se o klíčový nástroj pro správu a analytické účely v různých typech databázových systémů.

4.1 Databázové systémy

4.1.1 MySQL

Jedná se o open-source[16] relační databázový systém, který je velmi rychlý a snadno škálovatelný. Jeho přednosti jsou jednoduchost a snadné použití. Jak uvádí web MySQL (2023), systém je často využívaný pro webové aplikace a e-commerce platformy. Podporuje většinu běžných SQL příkazů a má vysokou dostupnost a spolehlivost.

4.1.2 PostgreSQL

Jde opět o open-source relační databázový systém s vysokou úrovní rozšiřitelnosti a flexibilitou. Tento systém podporuje rozšířenou funkcionalitu, jako jsou např. JSON datové typy a plně textové vyhledávání. Jak uvádí PostgreSQL Kolektiv (2023) na svém webu, systém je využíván pro enterprise aplikace a vysoké zatížení aplikací. Další vlastností je vysoká úroveň bezpečnosti a podpora vysoké dostupnosti.

4.1.3 Microsoft SQL Server

Tento proprietární relační databázový systém je vyvinutý společností Microsoft. Podle Microsoft Corporation (2023) jsou jeho přednosti snadná integraci s ostatními produkty Microsoftu, vysoká úroveň bezpečnosti a spolehlivosti a podpora velkých objemů dat. Dále podporuje různé programovací jazyky a obsahuje mnoho nástrojů pro řízení a správu dat.

[15] Structured Query Language
[16] Software s otevřeným zdrojovým kódem, který je technicky dostupný

4.1.4 Oracle Database

Obdobně jako u Microsoftu, jde o systém vlastněný společností Oracle. Jeho výhodami jsou podle Oracle Databases (2023) velké množství funkcí pro zpracování velkých objemů dat a vysoká úroveň škálovatelnosti. Je nejvíce využíván v podnikovém prostředí, protože zaručuje vysoké zabezpečení dat a vysokou dostupnost.

4.2 Funkce SQL

Tento jazyk nabízí mnoho funkcí pro manipulaci s daty v relačních databázích. Tyto funkce lze rozdělit do kategorií viz tabulky níže.

Tabulka 2: Manipulace s daty

Funkce	Akce
INSERT	Přidávání nových řádků do tabulky
UPDATE	Aktualizace existujících řádků v tabulce
DELETE	Smazání řádků v tabulce
SELECT	Výběr a filtrování dat z tabulky

Zdroj: vlastní zpracování

Tabulka 3: Definice databáze a tabulek

Funkce	Akce
CREATE DATABASE	Vytvoření nové databáze
CREATE TABLE	Vytvoření nové tabulky
ALTER TABLE	Úprava existující tabulky
DROP TABLE	Smazání tabulky

Zdroj: vlastní zpracování

Tabulka 4: Řízení transakcí

Funkce	Akce
COMMIT	Potvrzení transakce
ROLLBACK	Zrušení transakce
SAVEPOINT	Definice bodu, od kterého může být transakce zrušena

Zdroj: vlastní zpracování

Tabulka 5: Řízení uživatelů a oprávnění

Funkce	Akce
GRANT	Přidělení oprávnění uživateli
REVOKE	Odebrání oprávnění uživateli

Zdroj: vlastní zpracování

4.3 Trigger

Trigger (česky spouštěč) je v SQL specifický typ události, která je automaticky spuštěna, jakmile dojde k určité operaci na tabulce, např. při vložení, úpravě nebo odstranění záznamu. Funkce triggeru spočívá v tom, že může automaticky provádět určité akce na jiných tabulkách v rámci databáze, což umožňuje aplikovat více integrovaných úprav dat.

Triggery se často používají k validaci dat, kontrole integrity a zajištění správného výpočtu nebo tvorby výsledků. Například může být trigger nastaven na tabulku s objednávkami, aby automaticky kontroloval, zda je zákazník dostatečně starý na objednání produktu a zda má dostatek finančních prostředků na účtu. Pokud nedojde k splnění těchto podmínek, trigger může zrušit objednávku nebo přidat upozornění pro správce. Příklad triggeru je popsán níže.

```
CREATE TRIGGER uprav_log
AFTER INSERT ON objednavky
FOR EACH ROW
BEGIN
    INSERT INTO log (objednavka_id, action)
    VALUES (NEW.id, 'vlozeni');
END;
```

Tento trigger se zavolá po vložení nového záznamu do tabulky „objednavky“. Po vložení je

do tabulky „log“ vložen nový záznam s informacemi o nové objednávce.

4.4 Procedura

Procedura v jazyku SQL je specifický příkaz, který umožňuje uživateli definovat kroky, které se mají provést na úrovni databáze. Tyto kroky mohou zahrnovat manipulaci s daty, volání jiných procedur, řízení transakcí atd. Procedury mohou být volány v jiných příkazech SQL a jsou uloženy v databázi, takže mohou být opakovaně volány bez nutnosti opětovného definování. To umožňuje efektivnější využití kódu a snižuje pravděpodobnost chyb v důsledku opakovaného zápisu stejného kódu.

Procedury jsou také důležité pro zabezpečení dat, protože umožňují omezit přístup uživatele k datům pomocí specifických oprávnění. Tyto oprávnění mohou být nastavena na úrovni jednotlivých procedur, což umožňuje specifikovat, které funkce mohou být prováděny pomocí procedur. To pomáhá zabránit neoprávněnému přístupu nebo manipulaci s daty.

Výhodou procedur je také možnost jednoduchého a rychlého opravování a aktualizace kódu, protože stačí upravit pouze jedinou proceduru, která se opakovaně volá, a nikoliv každý příkaz SQL, který tuto proceduru volá.

Výsledkem je, že použití procedur v SQL zlepšuje efektivitu a zabezpečení aplikací, které využívají databáze. Příklad takové procedury je popsán níže.

```
CREATE PROCEDURE UpravInventar (@polozkaID INT, @mnozstvi INT)
AS
BEGIN
   UPDATE Inventar
   SET Mnozstvi = Mnozstvi + @mnozstvi
   WHERE PolozkaID = @polozkaID;
END
```

Tento příklad ukazuje, jak vytvořit proceduru v SQL, která přidá zadané množství produktu do inventáře. Argumenty @polozkaID a @mnozstvi se používají jako vstupy pro proceduru a update příkaz se používá k aktualizaci inventáře.

Tuto proceduru je možné volat např. pomocí příkazu níže, který přidá 10 kusů produktu s PolozkaID 5.

```
EXEC UpravInventar(5, 10)
```

4.5 Funkce

Funkce v SQL je pojmenovaný blok kódu, který provádí určitou operaci a vrací jednu

hodnotu. Funkce může být volána v rámci SQL dotazu, procedury nebo triggeru a jejím účelem je zpracování dat a výpočet hodnot. Funkce může obsahovat vstupní parametry a může vracet hodnotu libovolného datového typu.

Je tak užitečná např. při výpočtu agregačních funkcí nad daty, při manipulaci s daty a při validaci dat. Funkce mohou být také použity k vytvoření uživatelsky definovaných funkcí, které jsou speciálně navrženy pro konkrétní potřeby aplikace nebo databáze.

Pro přiblížení je příklad funkce popsán níže.

```
CREATE FUNCTION spocitej_prumer (IN id INT)
RETURNS FLOAT
BEGIN
	DECLARE celkem FLOAT;
	DECLARE pocet INT;
	SELECT SUM(cena), COUNT(*)
		INTO celkem, pocet
		FROM produkt
		WHERE id = id;
	IF pocet > 0 THEN
		RETURN celkem / count;
	ELSE
		RETURN NULL;
	END IF;
END;
```

Tato funkce přijímá vstupní parametr „id“ a vypočítá průměrnou hodnotu ze sloupce „cena“ tabulky „produkt“ pro zadané „id“. Pokud neexistují žádné záznamy s odpovídajícím „id“, funkce vrátí hodnotu NULL. Funkce může být použita k získání průměrné hodnoty pro konkrétní záznam v databázi viz příkaz níže, který vrátí průměrnou hodnotu ze sloupce „cena“ pro záznam s „id“ rovným 1.

```
SELECT calculate_average(1);
```

5 Metodika práce

5.1 Cíl práce

Hlavním cílem práce je analýza vybrané agendy podniku, tvorba konceptuálního a fyzického datového modelu. Dílčím cílem je implementace a otestování tohoto modelu na vybrané platformě a zhodnocení modelu.

5.2 Agenda

Analyzovanou agendou je skladové hospodářství malé nejmenované kamenické firmy z Jihočeského kraje. Pro pochopení kontextu, odkud se berou data a jakým způsobem v systému proudí informace, je provedena analýza několika vybraných byznys procesů a jsou znázorněny příklady případů užití (use case) v této firmě. Ke správnému definování procesů a use case je důležité specifikovat požadavky na systém. Na základě této analýzy je pak vytvořen model konceptuální, logický a fyzický. Tyto modely jsou vytvořeny v modelovacím nástroji, který je vybrán na základě vícekriteriálního hodnocení. Výsledná databáze je konzultována se subjekty z firmy, zda odpovídá požadavkům a bylo by možné jí nasadit.

5.3 Zdroj dat

Data vložená do vytvořeného modelu jsou čistě testovací a vznikla za účelem otestování funkčnosti modelu.

5.4 Testování

Fyzický model je vygenerován do SQL kódu, který je implementován do platformy bit.io, která je dostupná online. Datový model je v této platformě plněn daty a pomocí sady SQL příkazů je testována funkčnost modelu. Tyto příkazy pak mohou zastupovat funkcionality různých aplikací jako např. kliknutí na tlačítko.

6 Nástroje pro datové modelování

Aby bylo model možné vytvořit je nutné zvolit vhodný nástroj, jehož možnosti budou splňovat požadavky a potřeby pro tvorbu modelu. Vlastnostmi se podle Simplilearn (2022) rozumí např. podpora pro různé databázové platformy, vizualizace datových modelů, generování kódu, synchronizaci datových modelů. Jak dále udává DZone (2022), tak je dalším aspektem i cena licence, spolupráce a sdílení projektů a v neposlední řadě zabezpečení a oprávnění.

6.1 Vícekriteriální hodnocení bodovací metodou

V tabulce 2 je provedeno vícekriteriální hodnocení bodovací metodou pro výběr optimálního nástroje. Hodnoceny jsou ukazatele, kterými jsou podpora databázových platforem, generování DLL[17] souborů, naplnění databáze testovacími daty, podporované datové modely, cena licence nebo možnost využití zkušební verze.

Tyto ukazatele jsou subjektivně obodované na škále od 1 do 10. Následně je vypočten vážený průměr bodů, na jehož základě bude provedena volba nástroje.

V případě stejného počtu bodů rozhodují doplňující funkce, jako např. možnost generování dokumentace, podpora a aktualizace, podporované notace a možnost týmové práce.

[17] Data Definition Language (DDL) – soubor definující databáze a tabulky v ní viz kapitola 4.1

Tabulka 6: Výběr modelovacího nástroje

Ukazatel	Podpora databázových platforem	Generování DLL souborů	Naplnění databáze testovacími daty	Podporované datové modely	Cena licence / možnost využití zkušební verze	Průměr bodů
Váha	15 %	30 %	10 %	15 %	30 %	-
DbSchema	10	10	10	6	6	**8,2**
ER/Studio	9	10	8	10	10	**9,65**
ERWin Data Modeler	8	10	8	10	9	**9,2**
Toad Data Modeler	9	10	10	8	10	**9,55**
SQL Database Modeler	8	10	8	8	5	**7,7**

Zdroj: vlastní zpracování

Z tabulky 2 vyplývá, že bude použit nástroj ERStudio Data Architect od společnosti Idera Inc., jenž nabízí širší možnost tvorby logického a fyzického modelu a širší podporu databázových systémů na rozdíl od druhého umístěného Toad Data Modeleru.

7 Specifikace požadavků

Důležitým krokem v procesu vývoje systému je definování požadavků na systém, které zohledňují potřeby uživatelů daného systému. Takto definované požadavky by měly být podle Wiegerse & Beatty (2013) jednoznačné, přesně popsané, realizovatelné a měly by přinášet zúčastněným subjektům hodnotu.

Kamenictví se zaměřuje na výrobu parapetů, obkladů, kuchyňských desek, krbových obložení nebo náhrobků a pomníků. Model se zaměřuje na sektor výroby parapetů, který pokrývá více než třetinu výroby. Ostatní sektory jsou tomuto sektoru podobné jen s několika málo odchylkami. Ve výrobě jsou zpracovávané kamenné desky, z nichž se tyto výrobky vyřezávají, leští a impregnují. Tyto desky jsou dodávané ve standardním rozměru od dodavatele.

Organizační struktura firmy se skládá z ředitele, obchodního manažera, plánovače výroby a čtyř kameníků.

Stanovení požadavků je následující:

- Při dokončení zakázky potřebuje obchodní manažer vědět, zda byly u zakázky využity doplňkové materiály (stretch folie atp.).
- Jaký a kolik materiálu se skutečně spotřebovalo na zakázku a jaký byl odpad.
- Kamenná deska je unikátně označena. Ze zbytků desky lze vytvořit „novou“ desku a lze jí dále použít, pokud má požadované rozměry.
- Vlastnosti desky evidovat jak v metrech čtverečních, tak rozměrech, aby bylo možné rozeznat použitelnost desky pro další zakázku.
- Zarezervování materiálu zakázce, která bude realizovaná a tím odhalit případné nedostatky materiálu na skladu.
- Pokud z použité desky na zakázku již nebude možné nic využít, tak cenu odpadu přiřadit k dané zakázce.
- Pokud jsou nebo budou realizované zakázky se stejným materiálem, tak upozornit, aby tyto zakázky byly zpracovávány společně, kvůli ulehčení manipulace s materiálem.
- Zobrazení stavu zakázky online – příjem, řezání, leštění, impregnace, dokončení
- Získání informace kolikrát a kde byla využita jedna kamenná deska

7.1 Use case

Pro lepší porozumění potřebám uživatelů jsou provedeny příklady use case ke každému subjektu podniku. Jsou zaměřeny na to, co systém dělá z pohledu uživatele a jakým způsobem je s nimi propojen.

7.1.1 Zjištění podobné zakázky

Subjekt

Plánovač výroby

Popis

Plánovač otevře v systému zakázku; prohlédne si zakázky se stejným typem materiálu; dle rozměrů a dodacích lhůt zakázek rozhodne, jestli ji také předat do výroby či nikoliv.

Úspěšný proces

Plánovač najde další zakázku se stejným typem materiálu a rozhodne o přiřazení.

Basic flow

1. Otevře systém
2. Klikne na zakázky
3. Vyfiltruje si nepředané zakázky
4. Vybere zakázku
5. Klikne na tlačítko zakázky se stejným typem materiálu
6. Prohlédne si rozměry a dodací lhůty nabízených zakázek
7. Rozhodne o jejím přiřazení

7.1.2 Zanesení zbytkových formátů do systému

Subjekt

Kameník

Popis

Kameník otevře zakázku; načte desku pomocí čtečky, kterou použije; nařeže potřebné formáty; změří zbytky desky; ty, které mají méně než 0,09 m^2, zanese do systému jako

spotřebu k dané zakázce a přesune na vyhození; ostatní formáty zanese do systému, vytvoří nové štítky.

Úspěšný proces

Ze systému je vyskladněna původní deska a jsou zaneseny zbylé formáty.

Basic flow

1. Otevře denní výrobní plán
2. Klikne na zakázku
3. Zjistí rozměry a typ materiálu
4. Načte vyhovující desku
5. Převeze ji na pilu
6. Nařeže potřebné formáty
7. Změří zbylé kusy desky
8. Rozhodne o odpisu zbytku nebo o jeho novém zařazení do systému
9. Použitelnému zbylému kusu vytvoří nový štítek
10. Odepíše ze systému původní desku

7.1.3 Objednávka materiálu

Subjekt

Obchodní manažer

Popis

Obchodní manažer pomocí systému vyhodnotí nejprodávanější materiály; následně je přidá do hromadné objednávky, aby využil kapacitu objednaného kamionu.

Úspěšný proces

Obchodní manažer doplní objednávku materiálu nejprodávanějšími materiály.

Basic flow

1. Otevře systém
2. Otevře zakázky

3. Zobrazí prodaný materiál
4. Seřadí dle množství prodaného materiálu za posledních 6 měsíců
5. Zjistí stav na skladu nejprodávanějších materiálů
6. Zjistí stav objednávaného materiálu na přijaté zakázky
7. Vyhodnotí, zda přiobjednat nejprodávanější materiál do zásoby
8. Doplní objednávku o nejprodávanější materiály

7.1.4 Smluvení podmínek u dodavatelů

Subjekt

Ředitel

Popis

Ředitel si v systému vyhledá, od kterého dodavatele má firma největší odběr materiálu a pokusí se s tímto dodavatelem dojednat výhodnější obchodní podmínky.

Úspěšný proces

Ředitel dojedná výhodnější nákupní podmínky u největšího dodavatele.

Basic flow

1. Otevře systém
2. Otevře příjemky materiálu
3. Seřadí příjemky podle dodavatelů za posledních 12 měsíců
4. Provede součet hodnot jednotlivých příjemek dle dodavatelů
5. Zjistí největšího dodavatele
6. Domluví si s ním schůzku
7. Bude se snažit vyjednat lepší nákupní podmínky (vyšší slevu)

7.2 Procesy ve firmě

Pro přiblížení dalších informačních toků jsou zpracované byznys procesy, které ve firmě probíhají. Takové procesy vytváří komplexnější pohled na problematiku agendy a přibližují souvislosti v datech, s kterými tyto procesy pracují.

K analýze jsou vybrány 4 hlavní byznys procesy. Tyto procesy jsou zpracovány standardem BPMN. V jednotlivých procesech figurují subjekty, kterých se daný proces týká. Tyto subjekty tvoří tzv. dráhy, v kterých proces plyne a tyto dráhy spadají do tzv. bazénu, který znázorňuje prostředí, kde se proces odehrává. Pro lepší pochopení procesů následuje stručný popis s odkazem na příslušný proces.

V kapitole 7.2.1 je znázorněn proces vytvoření nabídky pro zákazníka, který začne tím, že zákazník projeví zájem o produkt a kontaktuje firmu se svými požadavky (např. jaký materiál, rozměry, kresba atp.). Firma následně tyto požadavky validuje a při nesrovnalostech se se zákazníkem vyrozumí. Po upřesnění požadavků je obchodním manažerem vytvořena cenová kalkulace, která je poslána ke schválení zákazníkovi. Ten nabídku může odmítnout s vyjádřením, které firma následně zpracuje. Pokud je vyjádření definitivní, proces končí. V případě některých nejasností firma zpracuje vyjádření a zašle novou nabídku. Po přijetí nabídky zákazník zašle firmě fakturační údaje a firma vystaví zálohovou fakturu, kterou zákazník následně zaplatí a proces končí.

V kapitole 7.2.2 je zpracován proces zadání zakázky, který začíná u plánovače výroby, který rozhoduje, která zakázka bude předána do výroby. Zjistí stav materiálu na skladu pro danou zakázku a v případě dostatku materiálu jej rezervuje pro zakázku a předá zakázku do výroby. V případě nedostatku musí materiál zajistit. V takovém případě vybere nasmlouvaného dodavatele, od kterého bude brát materiál a přidá materiál do budoucí objednávky. Aby se ušetřilo za dopravu, je výhodné naplnit celý kamion. Tudíž některé objednávky musí chvíli počkat, než se kamion naplní. Informaci o budoucí objednávce pošle dodavateli, který provede rezervaci materiálu a tuto rezervaci potvrdí. V momentě, kdy je kamion naplněn nebo je nutné provést objednávku, plánovač odešle objednávku dodavateli. Ten jí zpracuje a odešle. Následuje potvrzení od dodavatele a čekání na materiál. Po doručení je obchodním manažerem vytvořena příjemka a proces končí naskladněním materiálu.

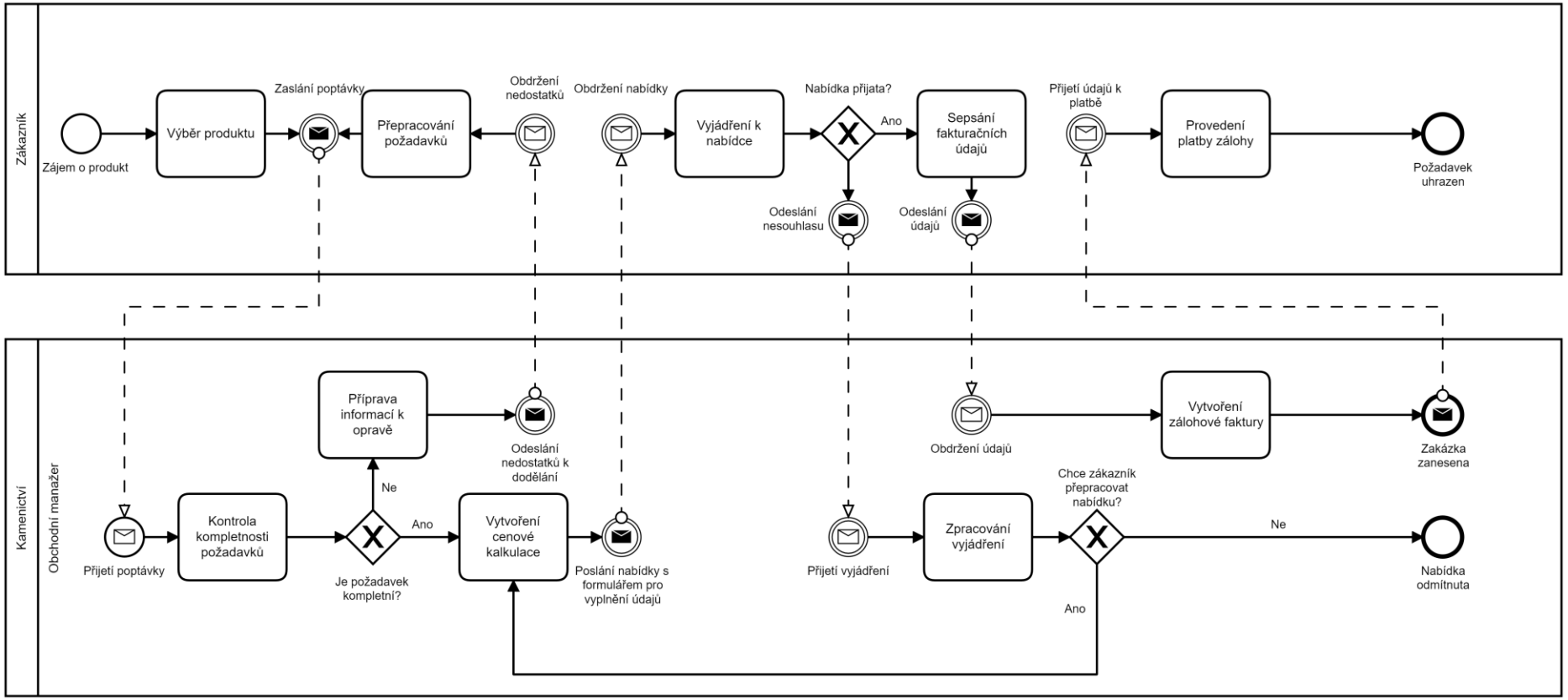
Další proces znázorněný v kapitole 7.2.3 je výroba zakázky. Ten začne přijetím zakázky od plánovače kameníkem, který má na starost řezání. Vybere materiál rezervovaný plánovačem a převeze jej na pilu. Následuje samotné řezání desky. Po dokončení řezání je deska předána na kolegy, kteří provedou leštění a impregnaci. Po dokončení těchto aktivit je deska zabalena s použitím voštiny a fólie a je uložena na sklad. Mezitím pilař po dokončení své práce naměří zbylé kusy materiálu z řezané desky a rozhodne, které kusy jsou již nepoužitelné a zadá je do odpadu, a které kusy jsou použitelné. Použitelné kusy zaeviduje znovu do skladu a proces je

ukončen.

Posledním znázorněným procesem je vydání zakázky v kapitole 7.2.4. Ten začíná dokončením zakázky. Následuje vytvoření výdejky obchodním manažerem a tvorbou faktury. Pokud bylo na začátku sjednáno předání osobně, tak obchodní manažer informuje zákazníka o dokončení jeho zakázky. Zákazník se osobně dostaví, provede platbu a je mu předána faktura. Zákazník si následně odveze desku a tím je obchod dokončen. Pokud byla sjednáno předání doručením, tak obchodní manažer kontaktuje zákazníka a domluví s ním termín dodání. Po smluvení termínu je obchodním manažerem odeslána faktura zákazníkovi a zároveň je sjednán dopravce, který desku doručí ve sjednaném termínu. Proces končí vydáním zakázky dopravci.

7.2.1 Proces vytvoření nabídky

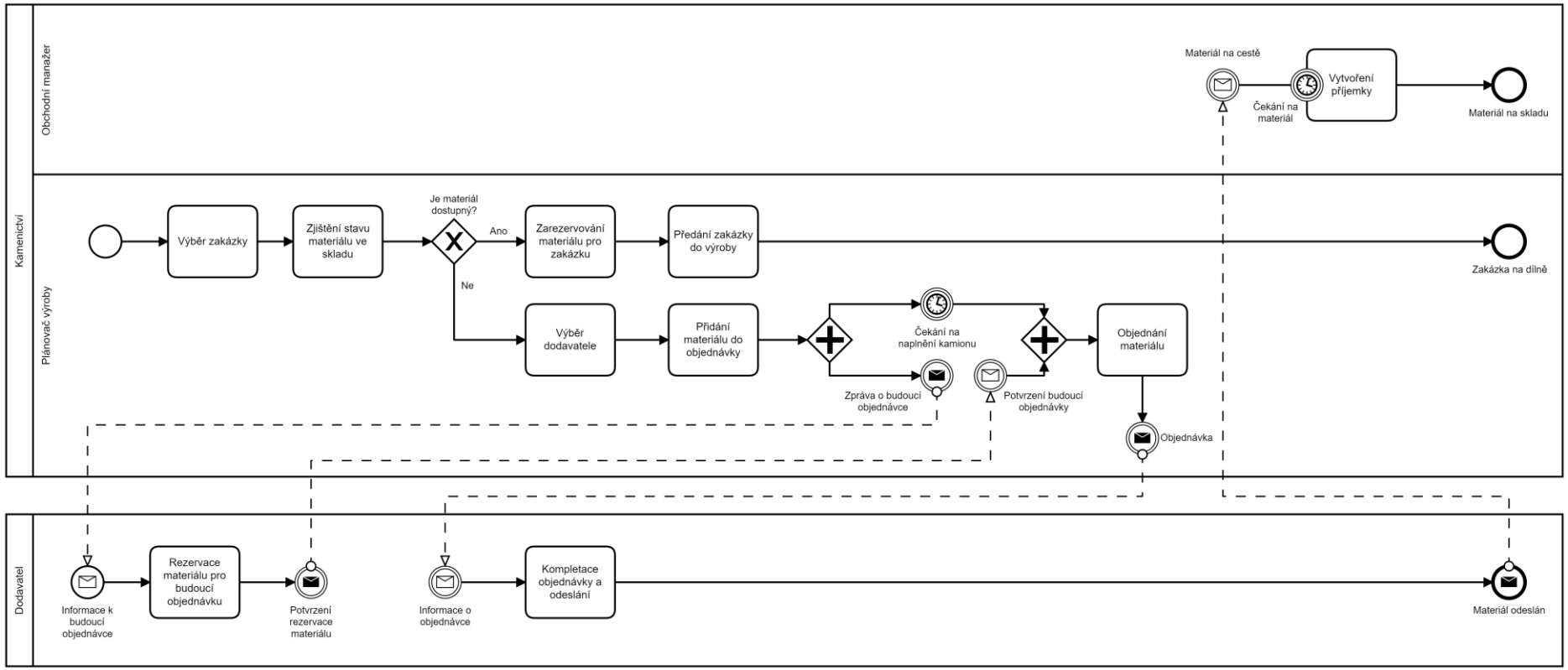
Obrázek 3: Proces vytvoření nabídky



Zdroj: vlastní zpracování v aplikaci Camunda Modeler

7.2.2 Proces zadání zakázky

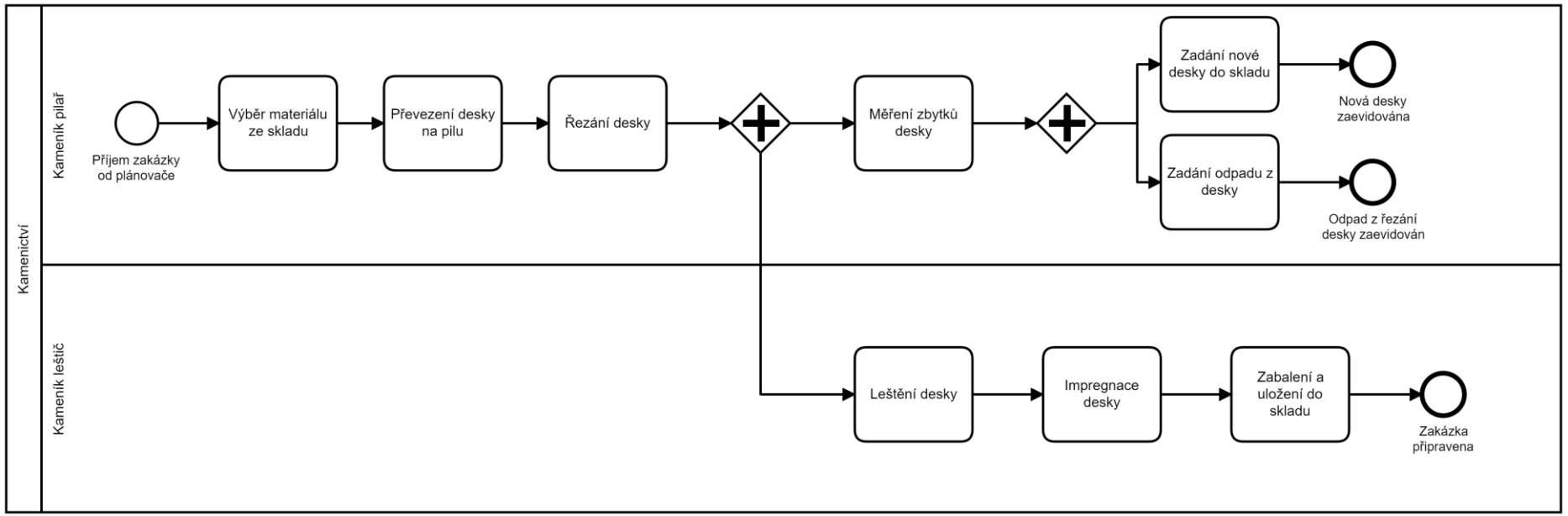
Obrázek 4: Proces zadání zakázky



Zdroj: vlastní zpracování v aplikaci Camunda Modeler

7.2.3 Proces výroby zakázky

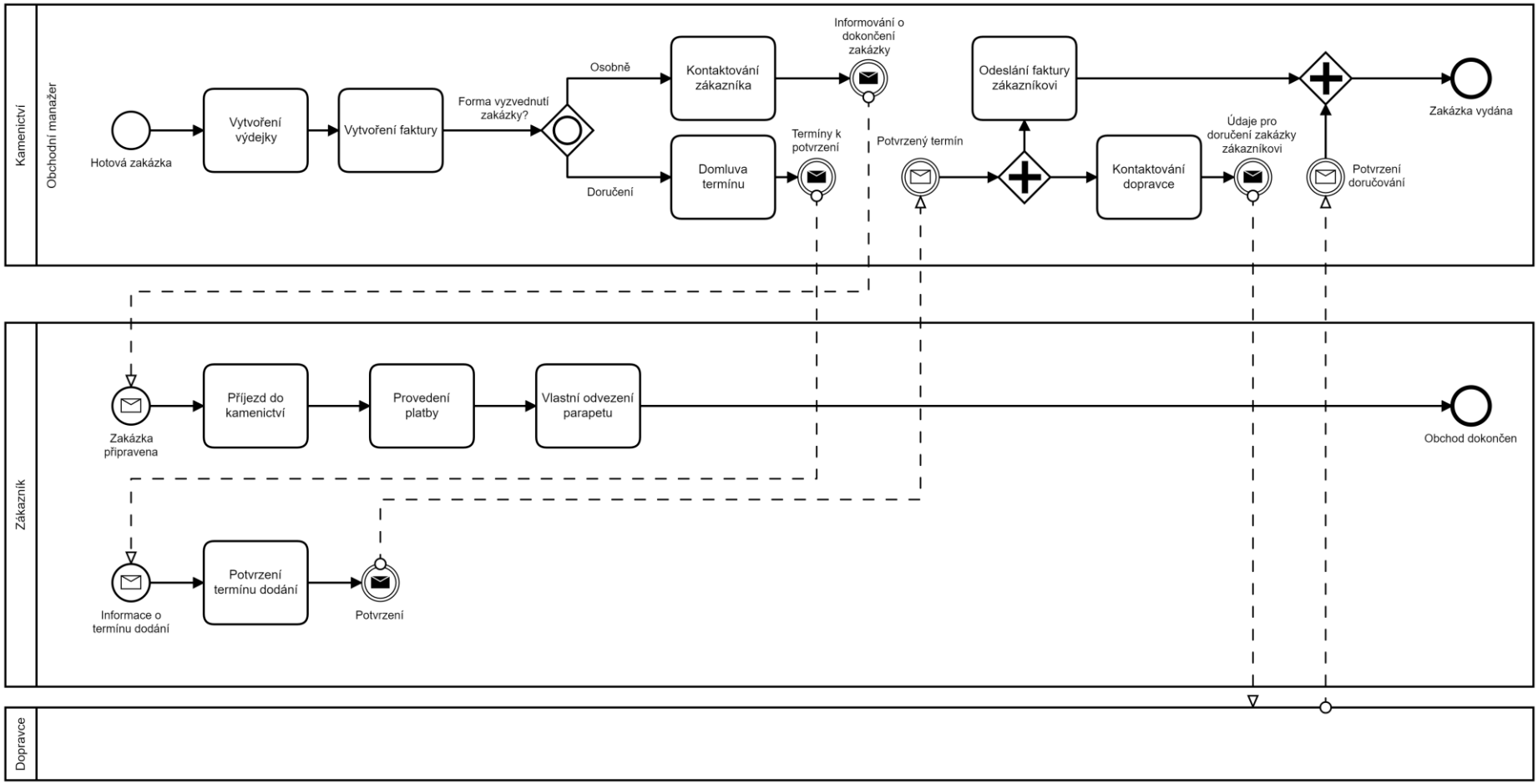
Obrázek 5:Proces výroby zakázky



Zdroj: vlastní zpracování v aplikaci Camunda Modeler

7.2.4 Proces vydání zakázky

Obrázek 6: Proces vydání zakázky



Zdroj: vlastní zpracování v aplikaci Camunda Modeler

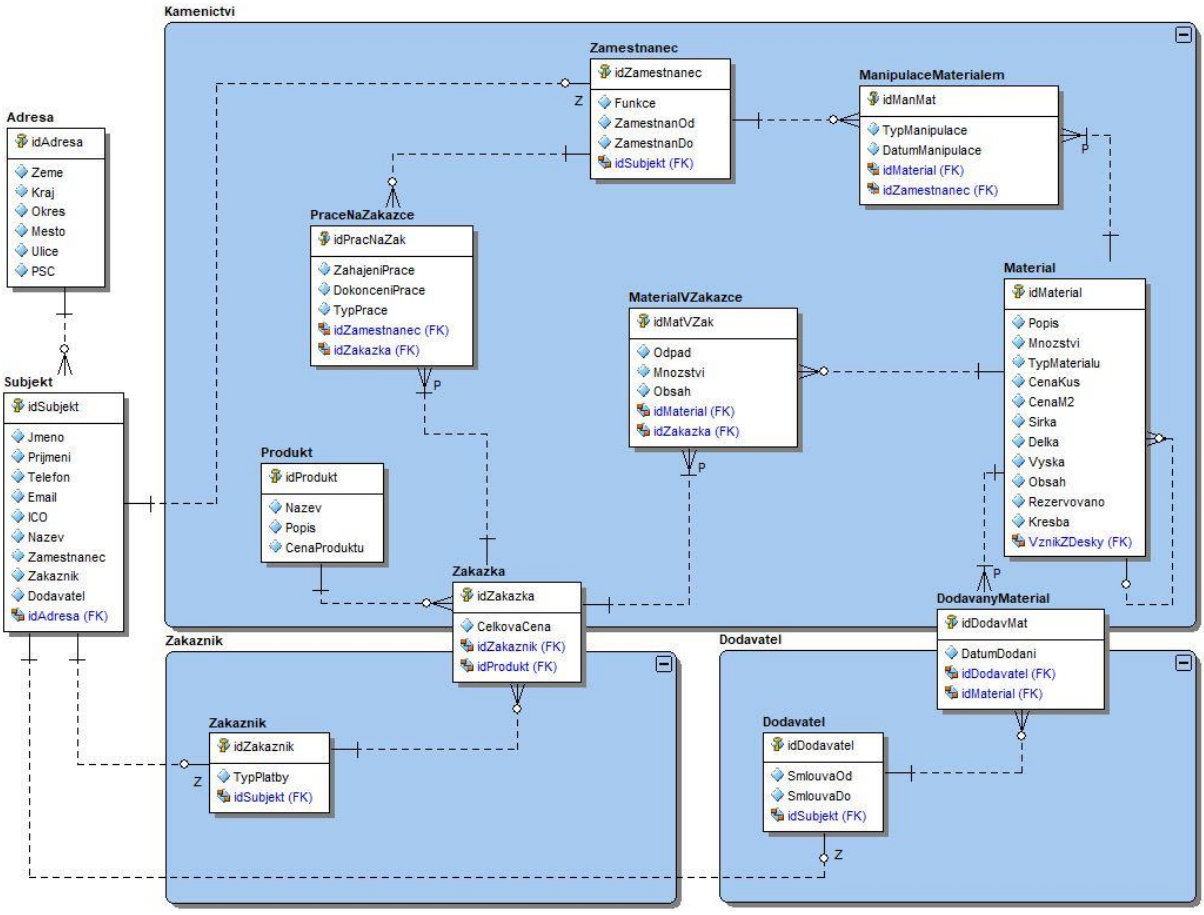
8 Tvorba modelu skladového hospodářství

Pro správnost a úplnost modelu jsou zohledněny požadavky z předešlé analýzy, které jsou strukturovány tak, aby byly jednoduché a přehledné. Nastavení entit, vztahů, atributů a jejich omezení je takové, aby byla práce s daty efektivní a bylo možné zpracovávat velké množství dat a zároveň, aby byla možná udržitelnost a rozšiřitelnost. Pro lepší integritu a konzistenci dat je model navržen ve třetí normální formě.

8.1 Konceptuální úroveň

Zpracování konceptuálního schématu v ERStudiu nabízí pouze tabulkovou notaci s atributy uvnitř a tabulky lze graficky zařadit do oblasti, do které data spadají konceptuálně viz modré oblasti na obrázku níže. Vztahy jsou již znázorněné vraními nohami. Dále je na obrázku 7 zobrazen konceptuální model jako celek, který je následně detailně vysvětlen. Jeho jednotlivé entity a jejich vlastnosti jsou popsány v kapitole 8.2. Zdrojový kód je uveden v příloze 1.

Obrázek 7: Konceptuální schéma kamenictví



Zdroj: vlastní zpracování v nástroji ERStudio Data Architect

Model obsahuje 8 nezávislých entit a 4 závislé entity upravující vztah M:N a definuje vztahy mezi nimi. Tzn. že jsou určené primární a cizí klíče. V modelu je vytvořena entita Subjekt, která je určena pro sjednocení informací o entitách zaměstnanců, zákazníků a dodavatelů. Vztah upravující tuto vazbu je jeden k jednomu nebo nule. Jedna instance subjektu tak může být přiřazena pouze jednomu zaměstnanci, zákazníkovi nebo dodavateli.

Entita PráceNaZakázce upravuje vztah mezi zaměstnancem a zakázkou. Zaměstnanec může a nemusí vykonávat práci na zakázce a zakázka na sebe musí mít vykázanou alespoň jednu práci.

Entita ManipulaceMateriálem definuje vztah mezi zaměstnancem a materiálem. Zaměstnanec může a nemusí manipulovat materiálem, ale materiál musí mít alespoň jednu manipulaci.

Entita DodávanýMateriál rozvádí vztah mezi dodavatelem a materiálem. Dodavatel může a nemusí dodávat konkrétní materiál, ale materiál musí být dodaný alespoň jedním dodavatelem.

Entita MateriálVZakázce upravuje vztah zakázky a materiálu. Zakázka musí obsahovat alespoň nějaký materiál, avšak materiál může a nemusí být v zakázce.

Entita Produkt tvoří instanci pouze pro zakázku a zakázka může, ale nemusí daný produkt obsahovat. Stejnou funkci má i entita Adresa. Ta se vztahuje k subjektu, kde k němu adresa může, ale nemusí být přiřazena.

K entitě Materiál se vztahuje rekurzivní vazba. Materiál může, ale nemusí obsahovat materiál, z kterého vznikl.

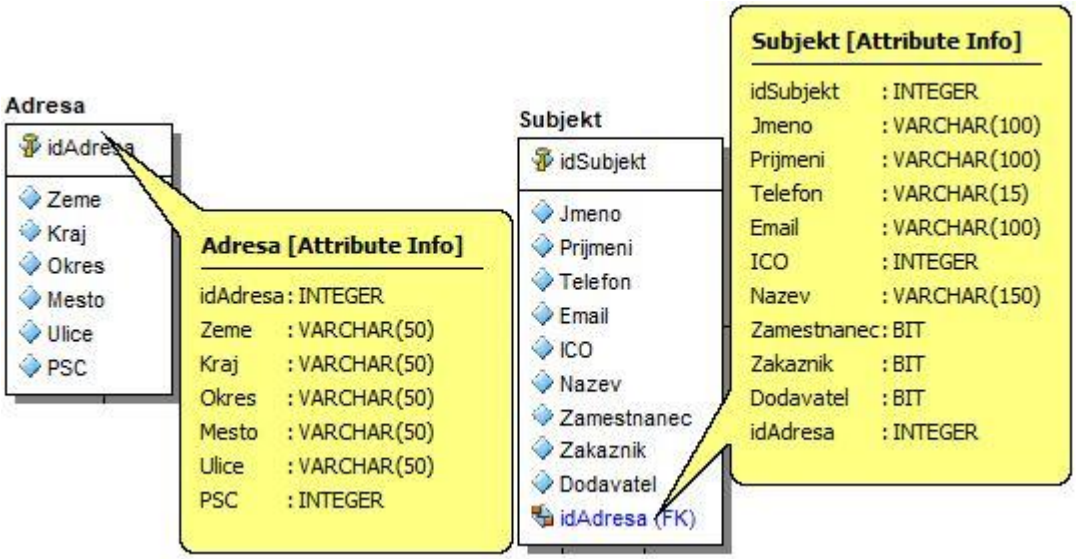
8.2 Logická úroveň

Jednotlivé tabulky v sobě mají vybrané atributy, které jsou zvolené na základě analyzovaných požadavků a dává tak smysl je uchovávat. Vlastnosti atributů jsou nastavené tak, aby zajišťovaly správnou validaci dat, zamezovaly vzniku nekonzistence dat v databázi a logicky podporovaly funkčnost celého systému.

Na obrázku 8 jsou zobrazeny tabulky Adresa a Subjekt. Subjekt má primární klíč určen atributem „idSubjekt“ a dále obsahuje mnoho kontaktních údajů, jako je jméno, příjmení, telefon, email. Dále uchovává informaci o tom, zda je subjekt zaměstnanec, zákazník nebo dodavatel v podobě logické proměnné. Tím lze dosáhnout rozlišení subjektů. Tato tabulka vznikla za účelem omezení redundance dat. Na cizí klíč u subjektu se váže tabulka Adresa

svým primárním klíčem „idAdresa“, ve které jsou převážně textové atributy definující adresu.

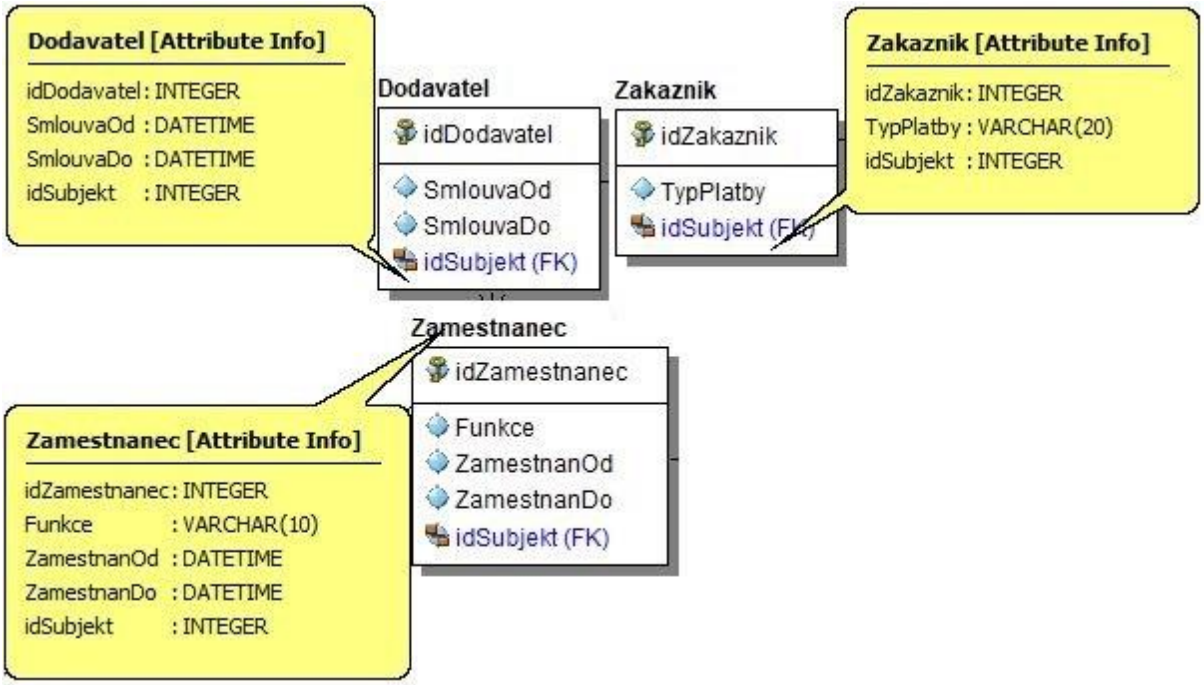
Obrázek 8: Tabulky Adresa a Subjekt



Zdroj: vlastní zpracování v nástroji ERStudio Data Architect

Tabulky, s kterými má subjekt vztah zobrazuje obrázek 9. Těmi jsou tabulky Dodavatel, Zákazník a Zaměstnanec. Každá tabulka má primární klíč stanovený unikátním ID. U dodavatele je uchováván pouze datum, kdy s ním byla sjednána nebo rozvázána smlouva. K zákazníkovi se vztahuje pouze atribut „TypPlatby“, který informuje o tom, jakým způsobem si zákazník při vyřizování zakázky domluvil předání a platbu. Zaměstnanec obsahuje atribut „Funkce“, který informuje o jeho pozici ve firmě. Dále tabulka obsahuje datumové atributy, které informují o délce vztahu zaměstnance se zaměstnavatelem. Ke všem těmto tabulkám je cizím klíčem navázána tabulka Subjekt.

Obrázek 9: Tabulky Zaměstnanec, Zákazník a Dodavatel



Zdroj: vlastní zpracování v nástroji ERStudio Data Architect

Na obrázku 10 jsou zobrazené tabulky Produkt a Zakázka. Obě mají primární klíč v podobě unikátního ID. Produkt obsahuje atributy, které uchovávají informace o nabízených produktech, které firma zpracovává. Produkt se váže na zakázku, která je sjednána zákazníkem. Tabulka Zakázka obsahuje atributy s cenou a cizí klíče z produktu a zákazníka.

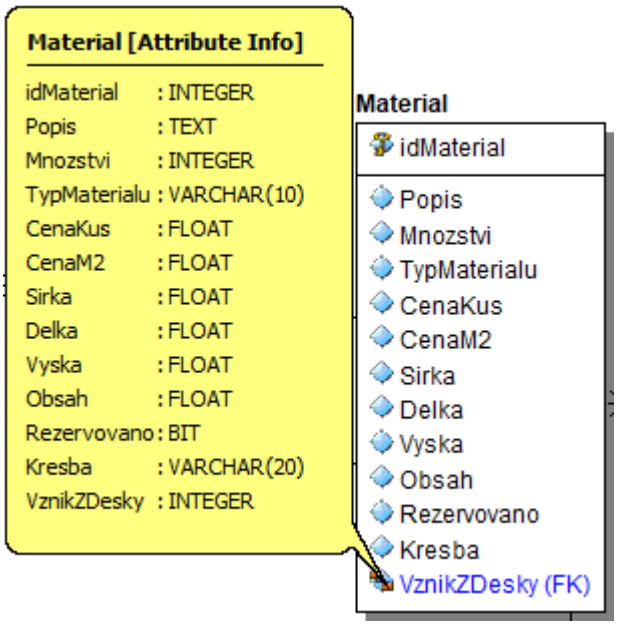
Obrázek 10: Tabulky Zakázka a Produkt

Zdroj: vlastní zpracování v nástroji ERStudio Data Architect

Tabulka Materiál, jež je zobrazena na obrázku 11, je velice důležitá. Obsahuje atributy podstatné pro splnění požadavků. Materiál je identifikován unikátním ID. Atribut „TypMaterialu“ rozlišuje, zda se jedná o výrobní materiál, tedy kamenné desky, nebo spotřební materiál, jako je folie či voština. Na základě tohoto rozlišení lze uchovávat rozměry

a cenu za m² u výrobního materiálu a cenu za kus a množství u spotřebního materiálu. Dalšími atributy u desek jsou kresba a informace, zda jsou rezervované. Atribut „VznikZDesky“ obsahuje informaci, zda deska byla vytvořená z jiné desky a slouží jako cizí klíč. Pokud byla, tak tento klíč odkáže na původní desku v té samé tabulce. U spotřebního materiálu tyto atributy nejsou podstatné a budou obsahovat hodnotu null.

Obrázek 11: Tabulka Materiál



Zdroj: vlastní zpracování v nástroji ERStudio Data Architect

Posledními a velice důležitými tabulkami jsou ManipulaceMateriálem, DodávanýMateriál, MateriálVZakázce a PráceNaZakázce. Všechny čtyři upravují M:N vztahy mezi nezávislými tabulkami a obsahují důležité informace např. o provedených operacích.

Tabulka ManipulaceMateriálem obsahuje atribut „TypManipulace“, který informuje, zda byl materiál přijat, vydán, vytvořen nebo rezervován a kdy byla tato akce provedena. Pomocí cizích klíčů z tabulek Zaměstnanec a Materiál je možná získat informaci, kterého materiálu se akce týká a kým byla provedena.

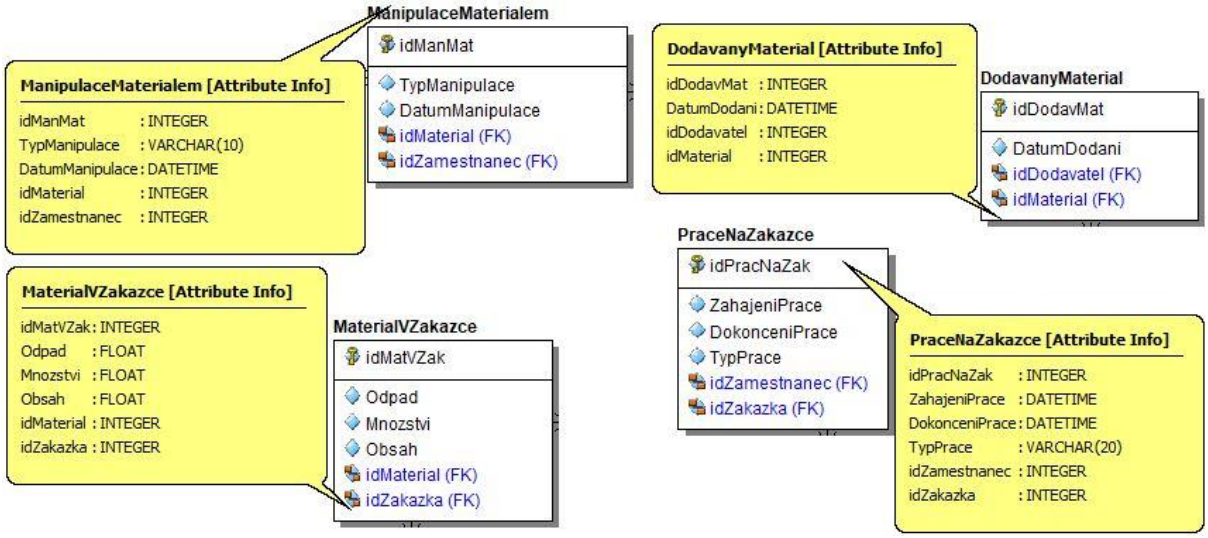
Z tabulky MateriálVZakázce lze získat informaci, které materiály byly použity do zakázky a lze je tak účtovat zákazníkovi. Atribut „Obsah“ informuje, kolik m² desky bylo použito na výrobu a „Odpad“ kolik z toho byl odpad. Atribut „Množství“ spadá spotřebnímu materiálu.

V tabulce PráceNaZakázce je atribut „TypPráce“, kde jsou uchovávány informace, jakou práci vykonával, jaký zaměstnanec. Jako např. nacenění, řezání, leštění, impregnace a dokončení. A tato práce je historizovaná datumovými atributy „ZahájeníPrace“

a „DokončeníPráce“. Tabulka obsahuje cizí klíče z tabulek Zaměstnanec a Zakázka.

Mezi Dodavatelem a Materiálem je poslední vztah upravující tabulka DodávanýMateriál. Ta obsahuje pouze atribut „DatumDodání“, ze kterého je možné získat informaci, kdy byl jaký materiál dodaný.

Obrázek 12: Tabulky ManipulaceMaterialem, DodávanýMateriál, PráceNaZakázce a MateriálVZakázce

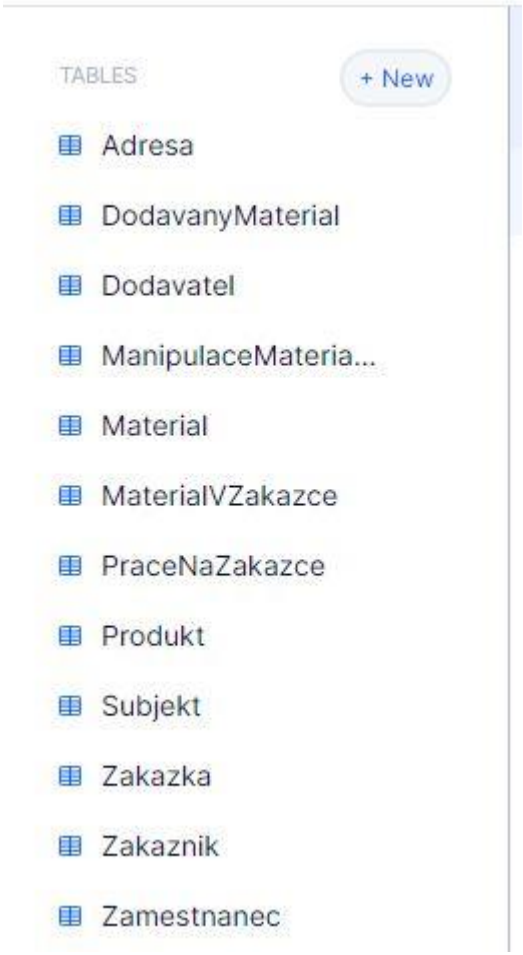


Zdroj: vlastní zpracování v nástroji ERStudio Data Architect

8.3 Nahrání fyzického modelu do platformy bit.io

Z vytvořeného logického modelu lze vygenerovat model fyzický a z něj lze získat SQL kód pro vytvoření databáze. Kód je generovaný pro databázový systém podporující PostgreSQL, který podporuje právě webová platforma bit.io. Součástí platformy je integrovaná příkazová řádka, jejíž prostřednictvím je model nahrán. Obrázek 13 dokazuje, že proces nahrávání byl úspěšný, protože databáze byla vytvořena. Včetně všech tabulek, atributů a omezení.

Obrázek 13: Nahraná databáze na platformě bit.io



Zdroj: vlastní zpracování z platformy bit.io

8.4 Testování databáze

8.4.1 Ošetření databáze

K ošetření proti chybám ze strany uživatele, jako třeba vložení špatného množství materiálu do zakázky, jsou vytvořeny pravidla, která se kontrolují při každém pokusu o úpravu záznamu. Tyto pravidla lze k vytvořenému modelu přidat prostřednictvím skriptu, nahraného do příkazové řádky platformy. V této kapitole je popsáno několik pravidel, které byly na modelu aplikovány.

Trigger pro kontrolu vložení a aktualizace hodnot v tabulce Materiál. Tento trigger zkontroluje, zda jsou vyplněny všechny požadované hodnoty a zda jsou číselné hodnoty v rozumných mezích u atributu „CenaKus“. Pokud některá z hodnot chybí nebo je neplatná, trigger zabrání vložení nebo aktualizaci záznamu. Pro samotnou kontrolu je vytvořena funkce

kontrolaCenaKus(), kterou trigger zavolá.

```
CREATE OR REPLACE FUNCTION kontrolaCenaKus() RETURNS TRIGGER AS $$
BEGIN
    IF NEW."CenaKus" < 0 THEN
        RAISE EXCEPTION 'Byla chybne zadana cena za kus';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS kontrolaCenaKus_trigger ON "Material";
CREATE TRIGGER kontrolaCenaKus_trigger
    BEFORE INSERT OR UPDATE ON "Material"
    FOR EACH ROW
    EXECUTE FUNCTION kontrolaCenaKus();
```

Pro podobnou kontrolu správně zadaných údajů jsou v modelu aplikovány další podobné triggery na další číselné atributy, které toto ověřují.

Další trigger je pro kontrolu zadávaného množství do tabulky MateriálVZakázce na základě dostupného množství v tabulce Materiál. Zadávané množství musí být rovno nebo menší než původní množství zadávaného materiálu.

```
CREATE OR REPLACE FUNCTION kontrolaPridanehoMnozstvi() RETURNS TRIGGER
AS $$
BEGIN
    IF TG_OP = 'UPDATE' AND OLD.“Mnozstvi“ - NEW.“Mnozstvi“ < 0 THEN
        RAISE EXCEPTION 'Nedostatek mnozstvi pro provedeni operace.';
    END IF;
    IF TG_OP = 'DELETE' AND OLD.mnozstvi < 0 THEN
        RAISE EXCEPTION 'Nedostatek mnozstvi pro provedeni operace';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER kontrolaPridanehoMnozstvi _trigger
    BEFORE UPDATE OR DELETE ON "Material"
    FOR EACH ROW
    EXECUTE FUNCTION kontrolaPridanehoMnozstvi ();
```

Proměnná TG_OP je v PostgreSQL speciální proměnná, která má návratovou hodnotu trigger. Pokud je trigger vytvořen pro kontrolu příkazů UPDATE nebo DELETE, tak se tato informace uloží do této proměnné.

8.4.2 Manipulace s daty

K otestování funkčnosti nastavených vztahů mezi tabulkami jsou využité příkazy pro

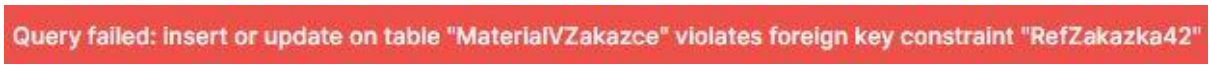
manipulaci s daty níže. Tyto příkazy mohou simulovat potvrzovací tlačítka v nějakém formuláři nebo aplikaci, kam by mohl být model implementován.

Posloupnost vkládání a spouštění příkazů je stejná, jak je vyobrazeno níže.

```
INSERT INTO "MaterialVZakazce"
      VALUES (2,null,null,null,3,1)
INSERT INTO "Material"
      VALUES (3, 'Exkluzivni mramorova deska', null , 'vyrobni' ,1,
400, 20, 50, 2, 1000, FALSE, 'Horizontalni', null);
INSERT INTO "Zakazka"
      VALUES (1,250000,1,1)
INSERT INTO "Produkt"
      VALUES (1,'Viceucelovy parapet','Parapet vhodny pro vsechny druhy
oken',20000)
INSERT INTO "Zakaznik"
      VALUES (1,'hotove',3);
INSERT INTO "Subjekt"
("idSubjekt","Jmeno","Prijmeni","Zakaznik","idAdresa")
      VALUES (3,'Karel','Nakupujici',TRUE,1)
INSERT INTO "Adresa" ("idAdresa","Mesto")
      VALUES (1,'Praha')
```

Při spuštění hned prvního příkazu dochází k chybě viz obrázek 14. To je zapříčiněno tím, že neexistuje zakázka, pro kterou je zde snaha vytvořit záznam. To potvrzuje dobře nastavené omezení databáze.

Obrázek 14: Chyba při pokusu vložit záznam



Zdroj: vlastní zpracování z platformy bit.io

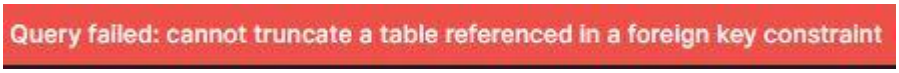
Tento případ nastane i při pokusu použít materiál, který neexistuje. Nejdříve je tak nutné vytvořit záznamy v nezávislých tabulkách. Ovšem v případě nastavování zakázky se opět zobrazí chyba. Pokud nebude založen zákazník, nemůže pro něj existovat zakázka. Toto se opakuje i u zákazníka, pro kterého musí být založen Subjekt. Po dodržení těchto omezení již s daty lze manipulovat.

V případě mazání dat je nutné otestovat referenční integritu. Příkaz níže se pokouší vyprázdnit tabulku Subjekt.

```
TRUNCATE TABLE "Subjekt" ;
```

Smazání dat však není povoleno viz obrázek 15. Daný subjekt je napojený na další tabulky a smazání by ohrozilo integritu dat.

Obrázek 15: Chyba při pokusu o smazání dat



Zdroj: vlastní zpracování z platformy bit.io

Pokud je však mazání nezbytné, lze použít klíčové slovo CASCADE a vymazání dat se pak promítne i do dalších tabulek a zajistí integritu.

8.4.3 Funkcionality

Pro dostání některým požadavkům lze nad daty v modelu provádět dotazy nebo procedury, které budou mít požadovaný výstup. V uživatelském prostředí by se takový dotaz nebo procedura uložila např. pod tlačítko, které by následně generovalo potřebné informace. Následuje demonstrace možných řešení pro některé požadavky specifikované v kapitole 7.

Řešení požadavku na zjištění stavu objednávky by vypadal následně.

```
SELECT TypPrace
      FROM PraceNaZakazce
WHERE idZakazka = *Požadované ID zakázky*;
```

Řešení požadavku na zjištění zakázek s podobným materiálem, které půjdou na pilu nebo na ní jsou by vypadal následně.

```
SELECT zak.idZakazka
FROM Zakazka as zak
INNER JOIN PraceNaZakazce prac
      on prac.idZakazka = zak.idZakazka
INNER JOIN MaterialVZakazce mat
      on mat.idZakazka = zak.idZakazka
WHERE prac.TypPrace in ('Naceneni','Rezani')
AND prac.ZahajeniPrace = *Požadovaný interval*
AND mat.idMaterial = *Požadovaný materiál*;
```

K ověření, zda byl zakázce přiřazen potřebný spotřební materiál, lze opět využít dotaz. Pokud ale bylo zadáno konkrétní množství tohoto materiálu do zakázky, je nutné toto množství upravit ve skladu. K tomu je využita následující procedura.

```
CREATE OR REPLACE PROCEDURE zadaniSpotrebnihoMaterialu(
material INT,
pocet INT,
zakazka INT)
LANGUAGE plpgsql
AS $$
DECLARE
p_posledni INT;
p_mnozstvi INT;
BEGIN
    SELECT "Mnozstvi"
    INTO p_mnozstvi
    FROM "Material"
    WHERE idMaterial=(material);
    SELECT max(idMatVZak)
            INTO p_posledni
            FROM "MaterialVZakazce";
    INSERT INTO
"MaterialVZakazce"("idMatVZak","Mnozstvi","idMaterial","idZakazka")
            VALUES (v_posledni+1, pocet, material, zakazka);
    UPDATE "Material"
            SET "Mnozstvi"="Mnozstvi"-pocet
            WHERE "idMaterial"=material;
end;
$$;
```

Při jejím provedení je uloženo množství spotřebního materiálu do tabulky MaterialVZakazce a toto přidané množství je odečteno z tabulky Material.

Při zadávání odpadu zakázce lze pomocí následující procedury spočítat cenu tohoto odpadu. Tuto cenu pak obchodní manažer připočte k celkové ceně za zakázku zákazníkovi.

```
CREATE OR REPLACE PROCEDURE vypocetCenyOdpadu(
IN p_material INT,
IN p_odpad FLOAT,
OUT p_cena_odpadu FLOAT)
AS $$
DECLARE
      p_cena_kus FLOAT;
BEGIN
--získání ceny kusu materiálu
      SELECT "CenaKus"
      INTO p_cena_kus
      FROM "Material"
      WHERE "idMaterial" = p_material;
-- výpočet ceny odpadu
      p_cena_odpadu := p_cena_kus * (p_odpad);
END;
$$ LANGUAGE plpgsql;
```

Model není navržený pouze pro splnění definovaných požadavků, ale lze na něm aplikovat i další funkcionality, které nebyli zahrnuty v požadavcích. Příklad takové funkcionality je uveden níže. Jde o nasazení konkrétní slevy na vybraný materiál. Kde sleva je vyjádřena procentuálně.

```
CREATE OR REPLACE PROCEDURE UpravitCenu(typ_materialu varchar(10),
koeficient float4) AS $$
BEGIN
    UPDATE "Material" SET "CenaKus" = "CenaKus" * koeficient
    WHERE "TypMaterialu" = typ_materialu;
END;
$$ LANGUAGE plpgsql;
```

Z výše uvedeného lze tvrdit, že k implementaci funkcionalit, které slouží k manipulaci s daty, je použití procedur optimální. Umožňují předdefinovat operace, které uživatelé mohou provádět nad daty a minimalizují tak riziko chyb a narušení dat. Proceduru lze rychle a snadno rozšířit o funkci v závislosti na potřebách uživatele nebo aplikace.

9 Zhodnocení a využitelnost datového modelu

Vytvořený datový model obsahuje náležitosti a omezení potřebné ke splnění stanovených požadavků. V případě potřeby úpravy nebo změny požadavků mohou být do modelu implementovány různé změny. Např. přidání atributů, změna omezení, přidání tabulek. Tyto úpravy lze provést pomocí reverzního inženýrství, které podporuje nástroj ERStudio Data Architect.

Model by dále mohl být využit jako rozšiřující doplněk nějakého ERP systému. V úvahu by mohl připadat např. ekonomický a účetní systém POHODA vyvíjený společností Stormware s.r.o. Ten nabízí rozhraní pro implementaci vlastního doplňku v jazyce C#. Výrobní podnik používající tento software by tak mohl propojit své účetnictví se skladovým hospodářstvím namodelovaným na míru.

Po konzultaci s obchodím manažerem z nejmenovaného kamenictví byl datový model shledán jako velmi přínosný. Doposud jsou veškeré informace mezi jednotlivými subjekty předávány pouze v písemné podobě nebo ústně. To vede k časové neefektivitě a vysoké pravděpodobnosti ztrátě dat a chybovosti. Firma má ambice rozšířit svoji působnost i za hranice Jihočeského kraje a zároveň obstát mezi místní konkurencí. Pokud by se ovšem navýšil objem zakázek, podnik se začne v datech ztrácet. Datový model by pomohl ulehčit práci především obchodnímu manažerovi a plánovači výroby. Dále by díky datovému modelu bylo možné generovat ekonomické reporty. Ty v tuto chvíli není možné provádět, protože obchodní manažer nemá potřebná data na jednom místě a bylo by velmi časově náročné tyto analýzy provádět.

Všechny ekonomické a účetní programy ve kterých lze vést skladové hospodářství nabízejí vést materiál pouze v metrech čtverečních. Tudíž datový model by byl užitečný pro všechna výrobní odvětví, která se zabývají i jednotlivými rozměry daného materiálu. Příkladem jsou truhlářství nebo krejčovství.

10 Závěr

Cílem práce je analýza vybrané agendy podniku a tvorba konceptuálního a fyzického datového modelu informačního systému pro tuto agendu na vybraném databázovém systému.

V analýze vybrané agendy jsou definovány požadavky na informační systém. Pro pochopení souvislostí a kontextu, jak by datový model měl vypadat, aby zachytil potřebnou realitu, jsou vytvořeny případy užití pro každý subjekt ve firmě. Dále jsou vyobrazeny důležité podnikové procesy ve standardu BPMN, které přibližují souvislosti mezi uchovávanými daty.

Pro tvorbu datového modelu je vybrán modelovací nástroj na základě vícekriteriálního hodnocení, kde kritéria a jejich váhy jsou voleny dle subjektivního úsudku autora. Vybraným nástrojem je namodelováno konceptuální schéma. V tom jsou zachycené potřebné entity a jejich atributy, aby bylo možné dostát definovaným požadavkům. Pro zajištění konzistence dat jsou vytvořeny vztahy mezi entitami. Z konceptuálního schématu je model převeden do logického ve třetí normálové formě, kde jsou definovány datové typy atributů. Následně je vytvořen model fyzický a je vygenerován skript pro vytvoření databáze.

Dílčím cílem práce je otestování modelu na webové platformě bit.io. Vytvořený datový model je zde nahrán pomocí vygenerovaného SQL skriptu pro databázový systém PostgreSQL 8.0. Jsou simulovány případy chybných manipulací s daty a vyhodnocováno nastavení proti těmto manipulacím. Dále jsou demonstrovány možnosti využití modelu k dosažení stanovených požadavků.

U malého podniku je vždy na majiteli, položit si otázku, zda se vyplatí přijmout dalšího pracovníka nebo investovat do koupě systému. Záleží vždy na konkrétní situaci a okolnostech. Koupě systému je z počátku nákladnou investicí, ale může být zároveň velmi rychle účinná a mít výsledky.

Summary

The objective of this thesis is to analyze a selected business agenda and create a conceptual and physical data model of an information system for this agenda on a selected database system.

The first part of the thesis are focused on description of information systems and their classification. Data modeling features and requirements are characterized in this part. Following part presents the selection of a modelling tool for creating a data model based on a multi-criteria evaluation. The conceptual schema is modeled using the selected tool. It captures the necessary entities and their attributes to meet the defined requirements.

The purpose of creating data model is capture the agenda of warehouse management of small stonemasonry company in South Bohemia region. Each chapter is focused on different part of creating database such as the conceptual, logical and physical levels. To test the created physical model SQL code is generated and uploaded to the web platform bit.io supporting PostgreSQL.

Keywords: information system, ERP, data model, BPMN, trigger, procedure, business process, SQL, database normalization, use case

Seznam literárních zdrojů

Aalst, W. (2016). *Process Mining: Data Science in Action*. https://doi.org/10.1007/978-3-662-49851-4

Beynon-Davies, P. (2003). *Database Systems* (3rd Edition). Red Globe Press.

Bruckner, T., Buchalcevová, A., Voříšek, J., Stanovská, I., Chlapek, D., & Řepa, V. (2012). *Tvorba informačních systémů*. Grada Publishing, a.s.

Buchalcevová, A. (2009). *Metodiky budování informačních systémů.* Oeconomica.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of Business Process Management*. Springer. https://doi.org/10.1007/978-3-662-56509-4

DZone. (2022, březen 3). *A Detailed Comparison of Data Modeling Tools—DZone*. DZone. https://dzone.com/articles/data-modeling-tools-comparison

Howe, D. (2001). *Data Analysis for Database Design* (3rd edition). Butterworth-Heinemann.

Chlapek, D., Kučera, J., & Palovská Helena. (2019). *Datové modelování a návrh relační databáze* (1.). Oeconomica.

Li, L., & Zhao, X. (2006). Enhancing competitive edge through knowledge management in implementing ERP systems. *Systems Research and Behavioral Science*, *23*(2), 129–140. https://doi.org/10.1002/sres.758

Li, Q., & Chen, Y.-L. (2009). Entity-relationship diagram. In *Modeling and Analysis of Enterprise and Information Systems* (s. 125–139). Springer.

Microsoft Corporation. (2023). *Microsoft Data Platform | Microsoft*. https://www.microsoft.com/en-us/sql-server

Miers, D., & White, S. A. (2008). *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies, Incorporated.

Muscatello, J. R., Small, M. H., & Chen, I. J. (2003). Implementing enterprise resource planning (ERP) systems in small and midsize manufacturing firms. *International Journal of Operations & Production Management*, *23*(8), 850–871. https://doi.org/10.1108/01443570310486329

MySQL. (2023). https://www.mysql.com/

Oracle Databases. (2023). *Cost-optimized and High-Performance Database*. https://www.oracle.com/database/

PostgreSQL Kolektiv, P. G. D. (2023, duben 5). *PostgreSQL*. PostgreSQL. https://www.postgresql.org/

QoChuk, B. (2021, září 4). *ER Diagram / Entity Relationship Model*. OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/entity-relation-model/

Rowley, J. (2007). The wisdom hierarchy: Representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180. https://doi.org/10.1177/0165551506070706

SAP. (2022). *Co je ERP | Definice plánování podnikových zdrojů | SAP Insights*. SAP. https://www.sap.com/cz/insights/what-is-erp.html

Simplilearn. (2022, duben 26). *Top 10 Powerful Data Modeling Tools You Should Know in 2023 | Simplilearn*. Simplilearn.com. https://www.simplilearn.com/top-powerful-data-modeling-tools-software-article

Simsion, G., & Witt, G. (2004). *Data Modeling Essentials*. Elsevier.

Tuteja, S. (2015, červenec 8). Normal Forms in DBMS. *GeeksforGeeks*. https://www.geeksforgeeks.org/normal-forms-in-dbms/

Wiegers, K., & Beatty, J. (2013). *Software Requirements*. Pearson Education.

Seznam obrázků, tabulek a příloh

Příloha 1: Datový model v SQL skriptu pro PostgreSQL 8.0

```
--
-- TABLE: "Adresa"
--
CREATE TABLE "Adresa"(
    "idAdresa"  int4          NOT NULL,
    "Zeme"      varchar(50),
    "Kraj"      varchar(50),
    "Okres"     varchar(50),
    "Mesto"     varchar(50),
    "Ulice"     varchar(50),
    "PSC"       int4,
    CONSTRAINT "PK12" PRIMARY KEY ("idAdresa")
)
;
--
-- TABLE: "DodavanyMaterial"
--
CREATE TABLE "DodavanyMaterial"(
    "idDodavMat"   int4        NOT NULL,
    "DatumDodani"  timestamp    NOT NULL,
    "idDodavatel"  int4        NOT NULL,
    "idMaterial"   int4        NOT NULL,
    CONSTRAINT "PK5" PRIMARY KEY ("idDodavMat")
)
;
--
-- TABLE: "Dodavatel"
--
```

```
CREATE TABLE "Dodavatel"(
    "idDodavatel"  int4         NOT NULL,
    "SmlouvaOd"    timestamp,
    "SmlouvaDo"    timestamp,
    "idSubjekt"    int4         NOT NULL,
    CONSTRAINT "PK6" PRIMARY KEY ("idDodavatel")
)
;
--
-- TABLE: "ManipulaceMaterialem"
--
CREATE TABLE "ManipulaceMaterialem"(
    "idManMat"         int4           NOT NULL,
    "TypManipulace"    varchar(10)    NOT NULL,
    "DatumManipulace"  timestamp      NOT NULL,
    "idMaterial"       int4           NOT NULL,
    "idZam"            int4           NOT NULL,
    CONSTRAINT "PK9" PRIMARY KEY ("idManMat")
)
;
--
-- TABLE: "Material"
--
CREATE TABLE "Material"(
    "idMaterial"    int4           NOT NULL,
    "VznikZDesky"   int4,
    "Popis"         text           NOT NULL,
    "Mnozstvi"      int4,
    "TypMaterialu"  varchar(10)    NOT NULL,
```

```
    "CenaKus"       float4,
    "CenaM2"        float4,
    "Sirka"         float4,
    "Delka"         float4,
    "Vyska"         float4,
    "Obsah"         float4,
    "Rezervovano"   boolean,
    "Kresba"        varchar(20),
    CONSTRAINT "PK4" PRIMARY KEY ("idMaterial")
)
;
--
-- TABLE: "MaterialVZakazce"
--
CREATE TABLE "MaterialVZakazce"(
    "idMatVZak"   int4      NOT NULL,
    "Odpad"       float4,
    "Mnozstvi"    int4,
    "Obsah"       float4,
    "idMaterial"  int4      NOT NULL,
    "idZakazka"   int4      NOT NULL,
    CONSTRAINT "PK3" PRIMARY KEY ("idMatVZak")
)
;
--
-- TABLE: "PraceNaZakazce"
--
CREATE TABLE "PraceNaZakazce"(
    "idPracNaZak"     int4          NOT NULL,
    "ZahajeniPrace"   timestamp,
```

```
    "DokonceniPrace"  timestamp,
    "TypPrace"        varchar(20)    NOT NULL,
    "idZam"           int4           NOT NULL,
    "idZakazka"       int4           NOT NULL,
    CONSTRAINT "PK7" PRIMARY KEY ("idPracNaZak")
)
;
--
-- TABLE: "Produkt"
--
CREATE TABLE "Produkt"(
    "idProdukt"     int4           NOT NULL,
    "Nazev"         varchar(100),
    "Popis"         text,
    "CenaProduktu"  float4,
    CONSTRAINT "PK11" PRIMARY KEY ("idProdukt")
)
;
--
-- TABLE: "Subjekt"
--
CREATE TABLE "Subjekt"(
    "idSubjekt"    int4           NOT NULL,
    "Jmeno"        varchar(100),
    "Prijmeni"     varchar(100),
    "Telefon"      varchar(15),
    "Email"        varchar(100),
    "ICO"          int4,
    "Nazev"        varchar(150),
    "Zamestnanec"  boolean,
```

```
    "Zakaznik"     boolean,
    "Dodavatel"    boolean,
    "idAdresa"     int4           NOT NULL,
    CONSTRAINT "PK10" PRIMARY KEY ("idSubjekt")
)
;
--
-- TABLE: "Zakazka"
--

CREATE TABLE "Zakazka"(
    "idZakazka"    int4      NOT NULL,
    "CelkovaCena"  float4    NOT NULL,
    "idZak"        int4      NOT NULL,
    "idProdukt"    int4      NOT NULL,
    CONSTRAINT "PK2" PRIMARY KEY ("idZakazka")
)
;
--
-- TABLE: "Zakaznik"
--
CREATE TABLE "Zakaznik"(
    "idZak"      int4          NOT NULL,
    "TypPlatby"  varchar(20),
    "idSubjekt"  int4          NOT NULL,
    CONSTRAINT "PK1" PRIMARY KEY ("idZak")
)
;
--
-- TABLE: "Zamestnanec"
```

```
--
CREATE TABLE "Zamestnanec"(
    "idZam"        int4          NOT NULL,
    "Funkce"       varchar(10)   NOT NULL,
    "ZamestnanOd"  timestamp,
    "ZamestnanDo"  timestamp,
    "idSubjekt"    int4          NOT NULL,
    CONSTRAINT "PK8" PRIMARY KEY ("idZam")
)
;
--
-- TABLE: "DodavanyMaterial"
--
ALTER TABLE "DodavanyMaterial" ADD CONSTRAINT "RefDodavatel72"
    FOREIGN KEY ("idDodavatel")
    REFERENCES "Dodavatel"("idDodavatel")
;
ALTER TABLE "DodavanyMaterial" ADD CONSTRAINT "RefMaterial82"
    FOREIGN KEY ("idMaterial")
    REFERENCES "Material"("idMaterial")
;
--
-- TABLE: "Dodavatel"
--
ALTER TABLE "Dodavatel" ADD CONSTRAINT "RefSubjekt322"
    FOREIGN KEY ("idSubjekt")
    REFERENCES "Subjekt"("idSubjekt")
;
--
-- TABLE: "ManipulaceMaterialem"
```

```
--
ALTER TABLE "ManipulaceMaterialem" ADD CONSTRAINT "RefMaterial282"
    FOREIGN KEY ("idMaterial")
    REFERENCES "Material"("idMaterial")
;
ALTER TABLE "ManipulaceMaterialem" ADD CONSTRAINT "RefZamestnanec292"
    FOREIGN KEY ("idZam")
    REFERENCES "Zamestnanec"("idZam")
;
--
-- TABLE: "Material"
--
ALTER TABLE "Material" ADD CONSTRAINT "RefMaterial152"
    FOREIGN KEY ("VznikZDesky")
    REFERENCES "Material"("idMaterial")
;
--
-- TABLE: "MaterialVZakazce"
--
ALTER TABLE "MaterialVZakazce" ADD CONSTRAINT "RefMaterial22"
    FOREIGN KEY ("idMaterial")
    REFERENCES "Material"("idMaterial")
;
ALTER TABLE "MaterialVZakazce" ADD CONSTRAINT "RefZakazka42"
    FOREIGN KEY ("idZakazka")
    REFERENCES "Zakazka"("idZakazka")
;
--
-- TABLE: "PraceNaZakazce"
--
```

```
ALTER TABLE "PraceNaZakazce" ADD CONSTRAINT "RefZamestnanec52"
    FOREIGN KEY ("idZam")
    REFERENCES "Zamestnanec"("idZam")
;
ALTER TABLE "PraceNaZakazce" ADD CONSTRAINT "RefZakazka62"
    FOREIGN KEY ("idZakazka")
    REFERENCES "Zakazka"("idZakazka")
;
--
-- TABLE: "Subjekt"
--
ALTER TABLE "Subjekt" ADD CONSTRAINT "RefAdresa342"
    FOREIGN KEY ("idAdresa")
    REFERENCES "Adresa"("idAdresa")
;
--
-- TABLE: "Zakazka"
--
ALTER TABLE "Zakazka" ADD CONSTRAINT "RefZakaznik92"
    FOREIGN KEY ("idZak")
    REFERENCES "Zakaznik"("idZak")
;
ALTER TABLE "Zakazka" ADD CONSTRAINT "RefProdukt332"
    FOREIGN KEY ("idProdukt")
    REFERENCES "Produkt"("idProdukt")
;
--
-- TABLE: "Zakaznik"
--
```

```
ALTER TABLE "Zakaznik" ADD CONSTRAINT "RefSubjekt302"
    FOREIGN KEY ("idSubjekt")
    REFERENCES "Subjekt"("idSubjekt")
;
--
-- TABLE: "Zamestnanec"
--
ALTER TABLE "Zamestnanec" ADD CONSTRAINT "RefSubjekt312"
    FOREIGN KEY ("idSubjekt")
    REFERENCES "Subjekt"("idSubjekt")
;
```