VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH DATABÁZE PRO SPRÁVU HARDWARE A SOFTWARE

DESIGN OF DATABASE FOR HARDWARE AND SOFTWARE ADMINISTRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ VÝMOLA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN LUHAN, Ph.D., MSc.
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Výmola Ondřej

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Návrh databáze pro správu hardware a software

v anglickém jazyce:

Design of Database for Hardware and Software Administration

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

BRUCKNER, T., J. VOŘÍŠEK, A. BUCHALCEVOVÁ a kol. Tvorba informačních systémů: Principy, metodiky, architektury. Praha: Grada Publishing a.s., 2012. 360 s. ISBN 978-80-247-4153-6.
CONOLLY T., C. BEGG a R. HOLOWCZAK. Mistrovství - Databáze : Profesionální průvodce tvorbou efektivních databází. Praha: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.
ELMASRI, R. and S. B. NAVATHE. Fundamentals of Database Systems. 6th ed. Boston: Addison Wesley, 2010. 1172 p. ISBN 978-0-136-08620-8.
SCHWALBE, K. Řízení projektů v IT: Kompletní průvodce. Praha: Computer Press, 2011. 632 s. ISBN 978-80-251-2882-4.

Vedoucí bakalářské práce: Ing. Jan Luhan, Ph.D., MSc

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc.	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 30.11.2015

ABSTRAKT

Předmětem této bakalářské práce je návrh databáze pro nadnárodní technologickou společnost. V databázi jsou sjednoceny informace o vlastnictví hardware a software. Standardní procesy jsou nastaveny pomocí procedur jazyka SQL. Tento proces značně zjednodušil práci uživatelů s informacemi. Díky tomuto návrhu se podařilo počet přístupů na server snížit na jednu třetinu oproti výchozímu stavu.

KLÍČOVÁ SLOVA

Databáze, SQL, relace, entita, hardware

ABSTRACT

Theme of this bachelor thesis is scheme of database for multi-national technological company. Information in database are merged information about ownership of hardware and software. Standard processes are set based on SQL language. This process highly made the work easier for users that work with information. Thanks to this design, it was easier to reduce amount of accesses to one third over the initial status.

KEYWORDS

Database, SQL, relation, entity, hardware

BIBLIOGRAFICKÁ CITACE

VÝMOLA, O. Návrh databáze pro správu hardware a software. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 46 s. Vedoucí bakalářské práce Ing. Jan Luhan, Ph.D., MSc.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně.

Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb. o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 19. ledna 2016 …….

Podpis studenta

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé diplomové práce, Ing. Janu Luhanovi, Ph.D., MSc. za jeho cenné rady, připomínky a čas, který mi věnoval při zpracování této bakalářské práce.

Poděkování patří také všem, kteří mi byli nápomocni při zpracování mé diplomové práce, především mým rodičům, Bc. Petru Kadlčíkovi a Ing. Tereze Mikulcové.

OBSAH

ÚVOD

Ve velkých nadnárodních společnostech, které vlastní obrovské množství majetku je obvyklé uchovávat jednotlivé položky v elektronických databázích, které jsou spravovány pracovníky IT oddělení. S počtem uživatelů roste i počet zařízení, proto je nutné IT oddělení rozdělit na menší týmy, pro správu hardware, softwarové specialisty a administrátory telefonů. Jednotlivé týmy si vytvářejí samostatné databáze pro jejich potřeby.

Tyto databáze jsou obvykle globálního charakteru. Při úpravách globálních databází lze narazit na velký problém v podobě obrovského množství entit a atributů, které zaručují fungování po celém světě, ale bohužel jsou kvůli regionálním zvyklostem a legislativě používány různě lokálními týmy. Databáze se stává stále větší a komplikovanější a ve výsledku to znamená zpomalení celého systému a tím zdržení pro koncového uživatele.

CÍL PRÁCE A METODIKA

Společnost využívá 3 různé databáze pro správu hardware a software. Cílem této práce je navrhnout jednu databázi, která nahradí ty aktuální. Jejím hlavním významem bude inventarizace hardware a software, dále bude sloužit evidenci vlastníků jednotlivých zařízení a licencí. Je nutné upravit aktuální procesy pro novou databázi, při zachování jejich logiky.

Dílčím cílem této práce je zjednodušit proces pro přiřazení zařízení pro nové uživatele ve společnosti, a zároveň zmenšit počet přístupů na databázi při odebírání zařízení od uživatele, který ukončuje svůj pracovní poměr.

Práce je uvedena teoretickým úvodem, který shrne problematiku datového modelování a databází. Tento úvod obsahuje i základy jazyka SQL, pomocí kterého je databáze navrhnuta. Následující část se zabývá analýzou aktuální situace ve společnosti a aktuálně nastavenými procesy. Tato kapitola také popisuje technické vybavení společnosti. Poslední část této práce obsahuje konkrétní návrh databáze, kde jsou využity teoretické poznatky z teoretické části. V této části lze najít i možná další rozšíření databáze, jejich důkladná studie však není součástí této práce.

1 TEORETICKÁ VÝCHODISKA

Celá bakalářská práce vychází z teoretických znalostí obsažených v této části. Dále kapitola obsahuje základy jazyka SQL a příklady jejich užití.

1.1 Základní pojmy

1.1.1 Data

Vychází z latinského slova *datum*, které lze přeložit jako *něco daného*. V kontextu informatiky se data používají pro označení čísla, textu nebo obrazu ve vhodné podobě pro zpracování počítačem.

Data reprezentují fakta a atributy, které bez dalšího popisu nemusejí dávat smysl. (10)

1.1.2 Informace

Jsou data uvedená do kontextu, která už jsou jasně srozumitelná a použitelná. Informace mohou znamenat pro různé lidi, v důsledku subjektivních znalostí a předpokladů založených na zkušenostech a názorech, různé věci. (10)

1.1.3 Entita

Množina objektů se společnými vlastnostmi, které uživatel označí jako nezávisle existující objekty. Entitou se může stát cokoliv, o čem je potřeba uchovávat nějaké informace. (1,2)

1.1.4 Atribut

Jedná o vlastnosti entity, resp. všechny důležité informace o entitě. Jednoduchý atribut, který se skládá pouze z jednoho člene, nazýváme atomický. Opakem je atribut složený, který se skládá z dvou a více členů. (1)

1.1.5 Relace

Mezi zúčastněnými entitami existují spojení. Množinu smysluplných spojení mezi těmito entitami nazýváme relace. Všechny relace musí být v rámci množiny jednoznačně identifikovatelné, tj. každá má svůj název, který popisuje jejich funkci. U každé relace se určuje její stupeň a kardinalita. Stupeň relace znamená, kolik atributů relace obsahuje. Kardinalita relace je počet entit v relaci. (1,3)

1.1.6 Primární klíč

Atribut nebo množina atributů, které jednoznačně identifikují všechny řádky relace. Primární klíč musí být jednoznačný, tj. žádná druhá entita v relaci nemá stejné hodnoty pro tuto množinu atributů. A zároveň žádný atribut není možné vypustit, aniž bychom porušili jednoznačnost, neboli primární klíč je minimální. (1)

1.1.7 Kandidátní klíč

Každá relace obsahuje více kandidátních klíčů, které všechny splňují požadavky primárního klíče, ale pouze jeden je vybrán jako primární klíč celé relace. Ostatní klíče se nazývají alternativní. (3)

1.1.8 Cizí klíč

Atribut relace, kde každá hodnota je buď úplně zadaná, nebo úplně nezadaná, a zároveň existuje jiná relace s primárním klíčem takovým, že zadaná hodnota cizího klíče je shodná s hodnotou primárního klíče nějaké entity této relace. (3)

1.2 Databáze

Je velké úložiště dat, které může být používáno více uživateli současně (1).

1.2.1 Normalizace

Normalizace je činnost, která datové struktury upravuje dle zvolené normy. Každá norma vychází z požadavku efektivního ukládání dat a minimální redundance. Aby byl model navržen optimálně, musí splňovat všechny normy:

- **První normální forma** – požaduje atomičnost, tj. všechny atributy entit jsou jednouché. Atomičnost atributů zvyšuje přehlednost databáze.
- **Druhá normální forma** – požaduje platnost první normální formy a závislost všech atributů na celém kandidátním nebo primárním klíči.
- **Třetí normální forma** – platí, pokud relace splňuje druhou normální formu, ve které jsou všechny hodnoty ve sloupcích závislé pouze na primárním klíči, a zároveň nejsou závislé na žádném jiném sloupci (1).
- **Boyce – Coddova normální forma** – požaduje platnost třetí normální formy a zároveň mezi kandidátními klíči není žádná funkční závislost, tj. relace musí mít

minimálně dva kandidátní klíče, minimálně dva z kandidátních klíčů jsou složené a kandidátní klíče se v některých atributech musí překrývat.

- **Čtvrtá normální forma** – platí, je-li relace v Boyce-Coddově normální formě a každá její vícehodnotová závislost je i funkční závislostí z kandidátních klíčů.
- **Pátá normální forma** – vyjadřuje cyklické omezení. Kde relace X je spojena s relací Y, relace Y je spojena s relací Z a relace Z je spojena s relací X. Pak platí, že všechny tři entity musí být součástí stejného vektoru hodnot (3).

1.2.2 Kardinalita vztahů mezi entitami

Mezi entitami existují různé vztahy. Porozumění těmto souvislostem mezi daty je velmi důležité pro implementování databáze.

Vztah jednoho k jednomu (1:1) - Existuje-li pro každý záznam entity A jeden záznam entity B a naopak.

Například: Jeden zaměstnanec má právě jeden telefon

Vztah jednoho k mnoha (1:N) – Jednomu záznamu entity A odpovídá více záznamů entity B a naopak

Například: Jeden počítač má jednoho majitele, ale majitel může mít více počítačů zároveň.

Vztah mnoha k mnoha (M:N) – Více záznamů entity A odpovídá více záznamům entity B a naopak.

Například: Jeden učitel učí více předmětů a stejný předmět učí více učitelů. (4)

1.3 Databázové modely

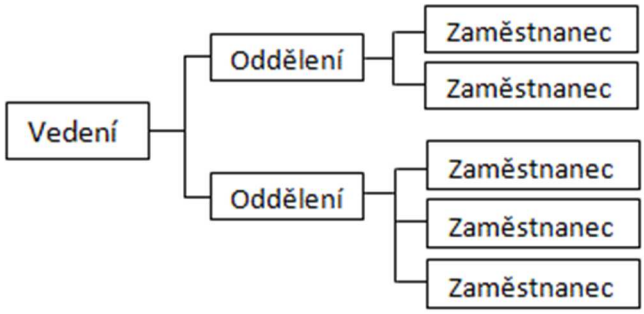
Databázový model je logický model, který se stará o reprezentaci dat.

Existuje několik databázových modelů. Nejznámějšími modely jsou hierarchický databázový model, síťový databázový model a speciální databázový model, který bývá často označován jako model relační. (4)

1.3.1 Hierarchický model

Hierarchický databázový model je nejstarší z aktuálně používaných modelů. Jedná se ospeciální případ síťového modelu. Diagram datové struktury tvoří strom nebo les stromů. Záznam může být součástí maximálně jednoho stromu. Typy záznamů jsou oproti síťovém modelu jednodušší a obsahují pouze jednoduché atributy. V hierarchickém

modelu se používají pojmy rodič (vlastník) a dítě (člen). Každý rodič může mít vztah k více potomkům, avšak každý potomek může mít vztah pouze k jednomu rodiči. (4,5)



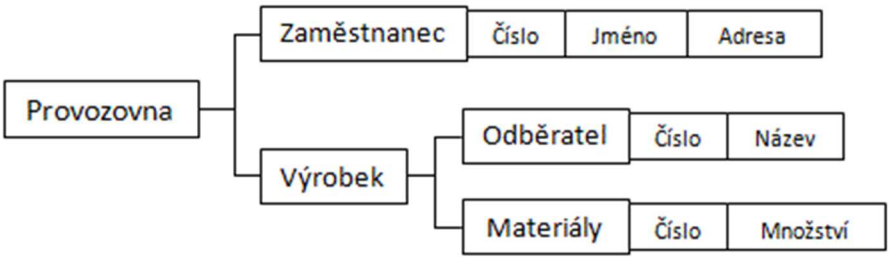
Obrázek 1: Hierarchický model (11)

1.3.2 Síťový model

Síťový databázový model byl vytvořen v roce 1971 skupinou DBTG (Data Base Task Group). Model je založen na souborech a vztazích mezi záznamy. Vztahy se tvoří pomocí spojek. Logický model databáze bývá označován jako schéma. Při definici schématu typ entity je nazýván jako typ záznamu.

Síťový model formuluje jen binární vztahy typů 1:1 a 1:N.

Největším nedostatkem tohoto modelu bývá obtížná implementace. Ač je síťový model pružnější než hierarchický, potíže s flexibilitou u něj stále přetrvávají, proto musí programátoři skvěle porozumět struktuře dat. V opačném případě model není výkonný. (4,5)



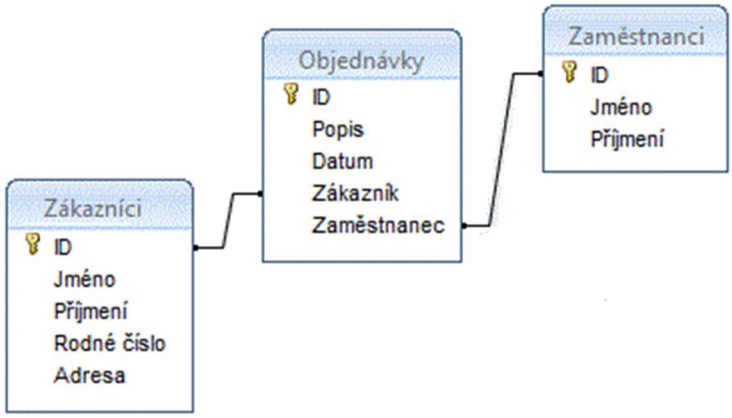
Obrázek 2: Síťový model (11)

1.3.3 Relační model

V roce 1970 zaměstnanec firmy IBM Dr. Codd popsal databázový relační model. Model je jednoduchý svou strukturou, data jsou organizována a uložena v tabulkách. Jednotlivé tabulky se skládají z řádků a sloupců. Všechny databázové operace jsou prováděny přímo s jednotlivými tabulkami.

Databáze vhodné pro relační model musí splňovat tyto vlastnosti:

- Databáze je chápána jako množina relací
- Jsou k dispozici operace selekce, projekce a spojení (4,5)



Obrázek 3: Relační model (11)

Dr. Codd definoval 12 pravidel pro relační databázi:

1. **Informační pravidlo** – Všechny informace v relační databázi musí být logicky popsány hodnotami v tabulkách.
2. **Pravidlo jistoty** – Všechna data v relační databázi jsou přístupná kombinací jméno tabulky, hodnota primárního klíče a název atributu.
3. **Systematické zpracování nulových hodnot** – Databáze plně podporuje nulové hodnoty, nezávisle na datovém typu.
4. **Dynamický on-line katalog založený na relačním modelu** – Metadata musejí být uložena stejně jako běžná data a musí k ním mít autorizovaní uživatelé přístup pomocí běžných relačních dotazů.
5. **Obsáhlý datový podjazyk** - Relační systém může podporovat několik jazyků a různých módů použitých při provozu terminálu. Nicméně musí být nejméně jeden příkazový jazyk s dobře definovanou syntaxí, který obsáhle podporuje definici dat, definici pohledů, manipulaci s daty interaktivně i programem, integritní omezení, autorizovaný přístup k databázi, transakční příkazy apod.
6. **Pravidlo vytvoření pohledů** – Všechny teoreticky možné pohledy je možné modelem vytvořit.
7. **Schopnost vkládání, vytvoření a mazání** – Schopnost zachování relačních pravidel je zachován při pohledech na data i při operacích přidání, mazání a průniku dat.

8. **Fyzická datová nezávislost** – Aplikační programy jsou nezávislé na fyzické datové struktuře.
9. **Logická datová nezávislost** – Aplikační programy jsou nezávislé na změnách v logické struktuře.
10. **Integritní nezávislost** – Integritní omezení se musí dát definovat prostředky relační databáze nebo jejím jazykem.
11. **Nezávislost distribuce** – Relační databáze musí být schopna implementace na jiných počítačových architekturách.
12. **Pravidlo přístupu do databáze** – Pokud relační systém používá jazyk nízké úrovně, nesmí být tato úroveň použita pro vytváření integritních omezení. Je nutno vyjadřovat se v relačním jazyce vyšší úrovně. (6)

1.4 Konceptuální návrh databáze

Na základě dat používaných uživateli je sestaven návrh databáze. Jeho cílem je vytvořit entity-relationship model (dále jen ER model), který musí splňovat požadavky na databázi. Postup návrhu se skládá z celkem devíti kroků, které postupně upřesňují ER model až do finální podoby. (1)

Identifikace entit – První krok definuje hlavní objekty databáze neboli entity. Je nutno je získat z požadavků uživatele na objekty a data uložené v databázi. Výsledkem identifikace je datový slovník, který obsahuje název, popis, předpokládaný počet výskytů a alias každé entity. (1)

Identifikace relací – V dalším kroku se určují vztahy mezi jednotlivými entitami. Zanesením modelu multiplicity je možné získat přesnější vyjádření relací v modelu, a zároveň je provedena kontrola všech dat. Výsledkem je datový slovník, popisující názvy entit, jejich vztahy a multiplicitu, popř. jednoduchý ER diagram. (1)

Identifikace atributů – V následující kroku se získají fakta o entitách, ze kterých jsou vytvořeny jejich atributy. Existují jednoduché, složené, vícehodnotové a odvozené atributy. Výstupem tohoto kroku by měla být tabulka s charakteristikou atributů, např. jméno, krátký popis, délka, alias, druh. (1)

Určení domén atributů – Je nutné získat množinu hodnot, ze kterých čerpají hodnoty jednotlivé atributy. (1)

Určení kandidátních klíčů a primárních klíčů – Po nalezení kandidátních klíčů v rámci entity, je nutno zvolit jeden primární klíč, ostatní klíče budou pouze klíči alternativními. (1)

Specializace entit – Nepovinný krok, ve kterém se modelují podtřídy a nadtřídy entit, které je potřeba od sebe odlišit. (1)

Kontrola redundance modelu – Přezkoumání všech vazeb v modelu. Nachází-li se v modelu vazba 1:1, tak při identifikaci entit došlo k přiřazení dvou entit témuž objektu. Jednu entitu je potřeba odstranit a zkontrolovat redundanci relací mezi entitami. (1)

Kontrola uživatelských transakcí – Ujištění se, zda model umožňuje transakce, které uživatel požaduje. Kontrola je prováděna samotným popisem transakce a sledováním cest transakcí. (1)

Diskuze nad konceptuálním návrhem s uživateli – V posledním kroku je nutno seznámit uživatele s výsledkem konceptuálního návrhu a zkontrolovat jeho požadavky. V tomto kroku jsou možné dodatečné úpravy a změny. (1)

1.5 Logický návrh databáze

Tato část vychází z konceptuálního návrhu ER diagramu. Relační tabulky se vytvářejí pomocí následujících 5 kroků:

Vytvoření tabulek – Výstupem je seznam tabulek, které jsou definovány jejich názvem, atributy a definovanými klíči.

Kontrola struktury tabulek – Kontrola třetí normální formy

Kontrola uživatelských transakcí – Kontrola požadavků uživatele na transakce

Kontrola integritních omezení – Logický návrh musí obsahovat všechna integritní omezení (NULL, omezení domén, referenční integrita apod.).

Uživatelská zpětná vazba logického návrhu – Návrhy na úpravy logického návrhu z pohledu uživatele. (1)

1.6 Fyzický návrh databáze

Fyzická implementace logického návrhu v prostředí cílové relační databáze.

Návrh je rozdělen do 6 kroků:

Převod logického návrhu databáze do cílového DBMS (Database management system) - Návrh tabulek a integritních omezení pomocí dostupné funkčnosti DBMS

Volba organizace souborů a indexů – Analýza transakcí, které jsou podporovány a výběr odpovídající organizace souborů a indexů

Návrh uživatelských pohledů – Rozhodnutí o implementaci všech uživatelských pohledů

Návrh bezpečnostních mechanismů – Návrh bezpečnostních opatření pro ochranu dat před neautorizovaným přístupem

Zvážení zavedení kontrolované redundance – Zváží se možnost uvolnění normalizace v logickém modelu dat, kvůli zvýšení celkové výkonnosti systému. Mohou se vyskytnout problémy s udržením konzistence dat

Monitorování a ladění systému v provozu – Pokračující monitorování a ladění systému v provozu, kvůli identifikaci, řešení problémů s výkonností a implementací změněných požadavků (1)

1.7 Jazyk SQL (Structured Query Language)

Jazyk SQL je strukturovaný dotazovací jazyk, který je standardizovaný a používaný pro práci s daty relačních databází. Vznikl v 70. letech ve firmě IBM pod názvem SEQUEL, při výzkumu relačních databází (3).

1.7.1 Základní dělení SQL

Jazyk pro definici dat

Tyto příkazy umožňují tvorbu, úpravu nebo smazání části databáze. Příkladem jsou to tyto příkazy (3):

- CREATE DATABASE – vytvoří databázi
- CREATE TABLE – vytvoří tabulku
- ALTER TABLE – upraví tabulku
- DROP TABLE – smaže tabulku

Jazyk pro manipulaci s daty

Tyto příkazy umožňují vybírat, vkládat, upravovat a mazat data ve vytvořených tabulkách. Jsou to tyto příkazy:

- SELECT – výběr dat
- INSERT – vložení dat
- DELETE – smazání dat

- UPDATE – úprava dat (3)

Jazyk pro správu práv

Tyto příkazy můžeme použít pro administraci práv uživatelů databáze. Používají se následující příkazy:

- CREATE USER – vytvoření uživatele
- ALTER USER – úprava uživatele
- DROP USER – smazání uživatele
- GRANT – přidělení privilegia (3)

1.7.2 Datové typy v SQL

Znakové řetězce:

- CHARACTER(n) – řetězec n znaků, max. 8000 znaků, prázdné znaky se doplňují zprava do velikosti řetězce
- VARCHAR(n) – řetězec n znaků, max. 8000 znaků, prázdné znaky se nedoplňují
- TEXT – stejně jako VARCHAR, max. 2GB

Číselné typy:

- TINYINT – 1 bajt
- SMALLINT – 2 bajty
- INTEGER – 4 bajty
- NUMERIC(n, m) – desetinná čísla, n čísel a m čísel za desetinou čárkou, 5-7 bajtů
- REAL – reálná čísla s plovoucí desetinnou čárkou, 4 bajty
- FLOAT(n) – reálná čísla s plovoucí desetinnou čárkou, 4 nebo 8 bajtů

Datum a čas:

- SMALLDATETIME – rrrr-mm-dd hh:mm:ss, 4 bajty
- DATE – rrrr-mm-dd, 3 bajty
- TIME – hh:mm:ss, 3-5 bajtů
- TIMESTAMP – struktura obsahující datum a čas (7)

1.7.3 Pokročilé databázové objekty

Pohledy – virtuální tabulky, které nemusí nutně v databázi existovat, ale generují se z tabulek databáze podle uživatelova požadavku. Jsou uloženy v databázi pro pozdější využití (1).

Procedury - všechny příkazy spuštěné v systému SQL můžeme uložit do procedury. Je to dávka příkazů jazyka SQL, která je pod svým názvem uložena do databáze. Volá se s požadovanými parametry. (8)

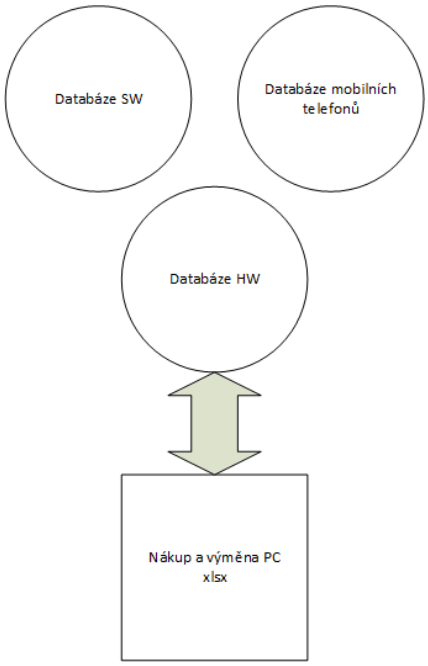
Spouště – cizím názvem triggery, jsou speciální procedury, které umožňují spouštět kód automaticky, při přidání, změně nebo smazání dat. (9)

2 ANALÝZA SOUČASNÉHO STAVU

Tato kapitola hodnotí aktuální situaci v nadnárodní technologické společnosti, způsob, jakým jsou uchovávána data o majetku a jak probíhají výměny nových počítačů. Dále je charakterizováno hardwarové vybavení společnosti.

2.1 Popis současné situace

Společnost pro evidenci HW používá jednu lokální databázi, která obsahuje všechny počítače, monitory, switche, televize, projektory a tiskárny. Tato databáze se nachází na databázovém serveru, kde je spravována databázovým administrátorem. Ve společnosti současně existují 2 globální databáze, kde první eviduje data o mobilních telefonech, jejich uživatelích a SIM kartách, druhá eviduje všechen SW, který byl nakoupen pro konkrétního uživatele. Obě tyto databáze jsou spravovány interně, ale bohužel mimo kompetence lokálního IT oddělení. Proto každá úprava databáze nebo její obnovení znamená velké časové zdržení. Protože jsou databáze používány po celém světě, je téměř nemožné prosadit jakoukoliv změnu procesu a každá změna si musí projít dlouhých schvalovacím procesem s nejistým výsledkem.
Mimo tyto databáze se používá tabulka v Excelu, která obsahuje data o všech PC, které je potřeba vyměnit a zároveň stav jejich výměny.



Obrázek 4: Databáze společnosti (Zdroj: vlastní)

2.2 Popis interních procesů technologické společnosti

Tato kapitola se zaměřuje na interní procesy technologické společnosti, které se odehrávají na IT oddělení a jsou spojeny s životním cyklem HW od jeho nakoupení po jeho ekologickou likvidaci a nákupem SW licencí.

2.2.1 Nákup HW

Pro nákup čehokoliv je ve společnosti potřeba schválení přímého nadřízeného a zároveň schválení vedoucího pracovníka, který je zodpovědný za rozpočet. Objednávka je zadána do interního elektronického obchodu, kde objednávku zpracují nákupčí a předají smluvně zavázaným dodavatelům. Po dodání si zboží IT pracovník vyzvedne na logistice, kde podepíše předávací protokol. Zboží vybalí, je-li potřeba dodatečná instalace, tak i nainstaluje. Vytvoří novou položku v databázi, přiřadí HW na majitele, který opět musí podepsat předávací protokol, kterým se zavazuje k odpovědnosti za daný předmět.

2.2.2 Nákup SW

Tento proces se velmi podobá nákupu HW, po objednání programu obvykle přichází mailové potvrzení o nákupu s odkazem, kde program stáhnout. Tento e-mail obsahuje i licenční číslo. K software není potřeba podepisovat žádný předávací protokol. Jakmile je software zakoupen, je spuštěn instalační skript, který jej nainstaluje na daný počítač.

2.2.3 Leasingová výměna PC

Leasingová výměna počítačů probíhá každý měsíc a její postup je následující. Na začátku měsíce se vygeneruje z účetního systému seznam všech zařízení, která se musí vrátit leasingové společnosti. Vygenerovaný seznam se uloží do souboru výměna PC. Zkontrolují se aktuální vlastníci počítačů podle databáze HW a ze seznamu se smažou položky, které byly vykoupeny z leasingové smlouvy.

Následuje zaslání oficiálního e-mailu ohledně výměny PC aktuálním majitelům a jejich přímému nadřízenému. Tento e-mail obsahuje název PC, jméno uživatele, datum, do kterého je potřeba počítač vrátit a navrhovaný typ nového PC i s možnými vylepšeními.

Nastávají 3 možné scénáře. Uživatel souhlasí se standardní výměnou, objednávka jde přímo na nákupčí. Uživatel požaduje změnit typ počítače nebo upravit konfiguraci, v tomto případě je vyžadováno schválení přímého nadřízeného a osoby zodpovědné za rozpočet. Uživatel už počítač nebude potřebovat, takže jenom vrátí starý. Po potvrzení nákupu počítače se postupuje podle standardního procesu nákup HW.
Po dokončení nákupu je počítač předán novému uživateli, který si převede svá data a starý počítač vrátí na IT oddělení. Paměť počítače je poté několikrát přepsána, aby se nikdo nedostal k datům, která obsahovala.
Po smazání je daný počítač zabalen a spolu se vším příslušenstvím odeslán zpět leasingové společnosti.

2.2.4 Převedení HW na jiného uživatele

Je-li potřeba převést počítač nebo jiný HW na jiného uživatele existují 2 možnosti:

- Převádí se v rámci jednoho týmu
- Převádí se do jiného týmu

V prvním případě stačí převést dané zařízení pouze v lokální databázi.
Je-li nutné zařízení převést na jiný tým, musí se vyplnit převáděcí formulář, který se pošle na účetní oddělení, které další leasingové platby bude účtovat na nového vlastníka.

2.2.5 Odkoupení nebo ztráta zařízení

Pokud je zařízení odkoupeno, tak je nutné zadat do systému poznámku, že zařízení bylo odkoupeno a nebude se vracet.
V případě ztráty, popř. krádeže zařízení se postupuje stejně, ale proces má pokračování na právním oddělení společnosti.

2.3 Hardwarové vybavení společnosti

Společnost disponuje zhruba 2000 počítači, většinu z nich dodala společnost Dell, od které je zboží zapůjčeno pomocí operativního leasingu. Na každý z nich je možno uplatnit „Next business day“ záruku na 3 roky. Servisní technik dorazí závadu opravit na místo určení do 24 hodin od nahlášení závady.

Projektory a ostatní příslušenství je taktéž zapůjčeno pomocí operativního leasingu a platí pro ně stejné záruční podmínky jako pro počítače.

Dell monitory jsou nakupovány společností se zárukou 5 let.

Aktuálně společnost disponuje těmito typy HW:

- Desktopy
    - Optiplex 790
    - Optiplex 7010
    - Optiplex 3020
    - Precision T1500
    - Precision T3500
    - Precision T5500
    - Precision T7910
- Laptopy
    - Latitude E6420
    - Latitude E6430
    - Latitude E6230
    - Latitude E5440
    - Latitude E7440
    - Latitude E7420
    - Precision M4800
    - Precision M6800
- Projektory
    - Dell 1610HD
    - Dell 4220
- Monitory
    - Dell P2214
    - Dell P2414
    - Dell U2413
- Příslušenství
    - Myš a klávesnice Dell
    - Dell Advanced E-Port II
    - Zámek Kensington MicroSaver

Společnost disponuje 95 zasedacími místnostmi, které jsou vybaveny LCD televizí nebo projektorem. Místnosti, které mají víc jak 10 míst na sezení, mají uvnitř videokonferenci Polycom, která obsahuje 360° webkameru.

Síť je spravována uvnitř patch-roomů, do kterých je vedena síť ze serverovny do switchů. Do nich se připojují jednotlivé zásuvky. Patch-roomů je 12, tj. cca 60 switchů od společnosti Cisco.

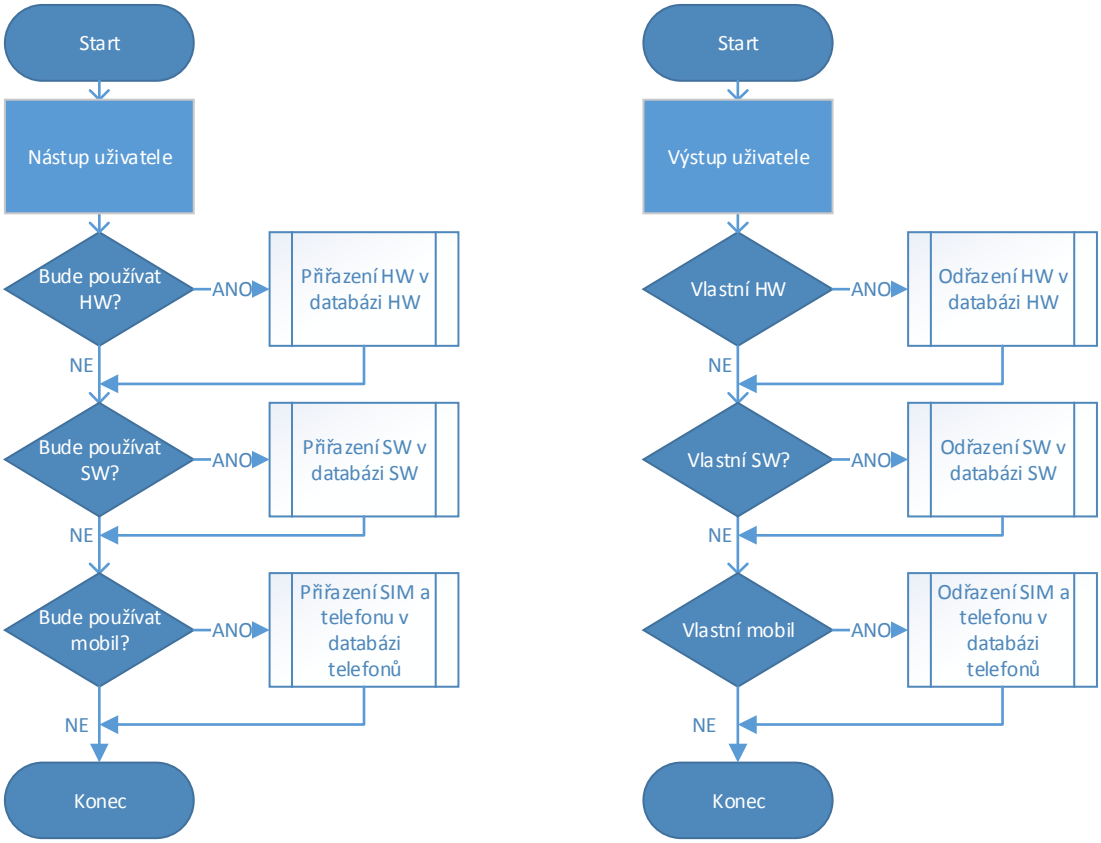
Tiskárny jsou vypůjčeny od společnosti XEROX. Platí se paušálně podle počtu tiskáren. Servis a spotřební zboží je v ceně paušálu. Každá tiskárna má k sobě připojeno zařízení SafeQ, pomocí kterého je možné tisknout kdekoliv v budově na vlastní čipovou kartu. Společnost disponuje 20 tiskárnami XEROX Workcentre 7835 a 5 tiskárnami XEROX Workcentre 7335.

Většina zařízení běží na systému Windows 7. Většina uživatelů používá programy pro běžnou kancelářskou práci jako například Microsoft Office 365, Adobe Acrobat, Total Commander, Xmind. Na výpočetních stanicích jsou instalovaný různé výpočetní modely nebo programy pro 3D modelování na bázi CAD.

2.4 Hodnocení aktuální situace

Na IT oddělení společnosti je používáno několik procesů. Časově nejnáročnější jsou proces pro nástup uživatele a výstup uživatele. V těchto případech je nutné otevřít každou databázi na oddělení, uživatele manuálně přidat a přiřadit mu zařízení, která bude používat, v opačném případě uživatele zablokovat a zařízení přiřadit na sklad nebo do inventáře.

Z diagramu lze vyčíst, že při každém nástupu a výstupu uživatele jsou nutné 3 přístupy na 3 různé databáze, dále vlastní přiřazení a odebrání zařízení.



Obrázek 5: Proces Nástup a Výstup uživatele (Zdroj: vlastní)

2.5 Požadavky na databázi

Navrhovaná databáze bude sloužit výlučně pro potřeby IT oddělení, a to pro evidenci majetku a jeho vlastnictví. Databáze by měla uchovávat pouze aktuální data, protože důležitý je pouze aktuální stav. Do databáze budou zařízení přidávat zodpovědní zaměstnanci IT oddělení ihned při převzetí zboží. Po vrácení zařízení, popř. jeho likvidaci se zařízení nastaví stav „vráceno“, tj. záznam o zařízení bude existovat, ale bude neaktivní. Jedním z požadavků bylo databázi sjednotit a zjednodušit oproti aktuálnímu stavu. Databáze by měla obsahovat maximálně deset tisíc záznamů.

Cíle zjednodušit přiřazování zařízení novým uživatelům bude dosaženo už přenesením aktuálních dat do jedné databáze, tím zkrátíme počet přístupů do databáze ze tří na jeden přístup.

Bude nutné naprogramovat trigger nad databází, který zařízení uživatele, který ukončuje pracovní poměr, převede na sklad a umožní jeho následné přiřazení.

3 VLASTNÍ NÁVRH ŘEŠENÍ

Tato část obsahuje navrhované řešení analyzovaného problému. Na začátku se identifikují všechny entity, které databáze obsahuje a relace mezi nimi. Konceptuální návrh je dokončen vytvořením ER diagramu. Následuje seznam všech tabulek z hlediska jejich využití a klíčů v logickém návrhu. Další část řešení zahrnuje samostatné vytvoření tabulek, pohledů, procedur a spouští v prostředí Microsoft SQL Server 2008. Poslední část je zaměřena na využití navrhovaného modelu v praxi a jeho každodenní používání zaměstnanci na IT oddělení.

3.1 Administrace databáze

Přístup k databázi je možný přes webový prohlížeč pro všechny zaměstnance společnosti. Při zobrazení webového prostředí se zobrazuje pohled podle kategorie uživatele. Kategorie jsou následující:

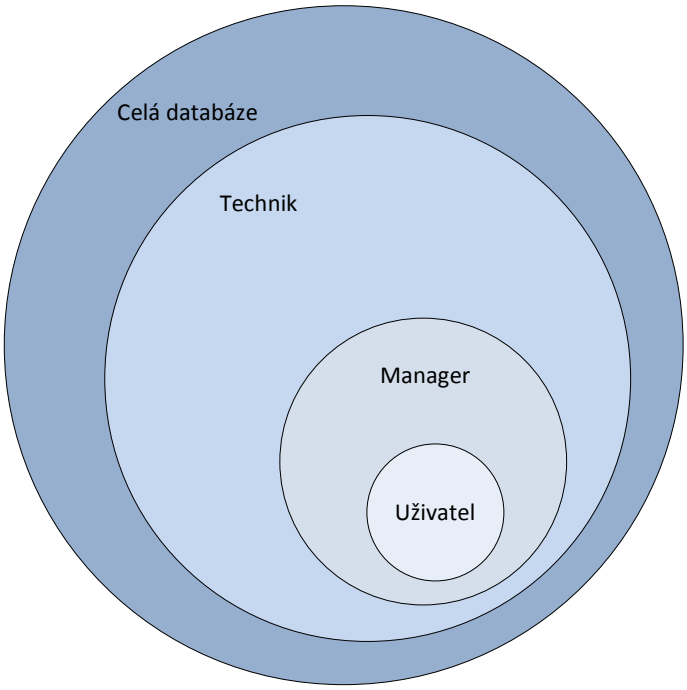
- Administrátor
- Technik
- Manažer
- Uživatel

Administrátor je vlastníkem databáze, kterou spravuje a udržuje aktuální. Má jako jediný úplný přístup k fyzickým datům, která může upravovat.

Technik je schopen přes webové rozhraní najít všechna aktivní zařízení a jejich vlastníky. Má práva na úpravu jednotlivých atributů a změnu vlastníků všech zařízení.

Manažer podobně jako technik má práva na změnu vlastníků zařízení. Naopak pro něj není aktivní možnost změny atributu a jeho pohled na databázi je značně ořezán. Pod profilem manažera se zobrazují pouze zařízení, která vlastní všichni jeho podřízení.

Uživatel může na webu zobrazit pouze svá zařízení, která však nemůže nijak změnit.
Pohledy na databázi jednotlivých kategorií uživatelů lze zobrazit obrázkem, jejich konkrétní implementace je obsažena nastíněna ve fyzickém návrhu:



Obrázek 6: Pohledy na databázi (Zdroj: vlastní)

3.2 Procesy databáze

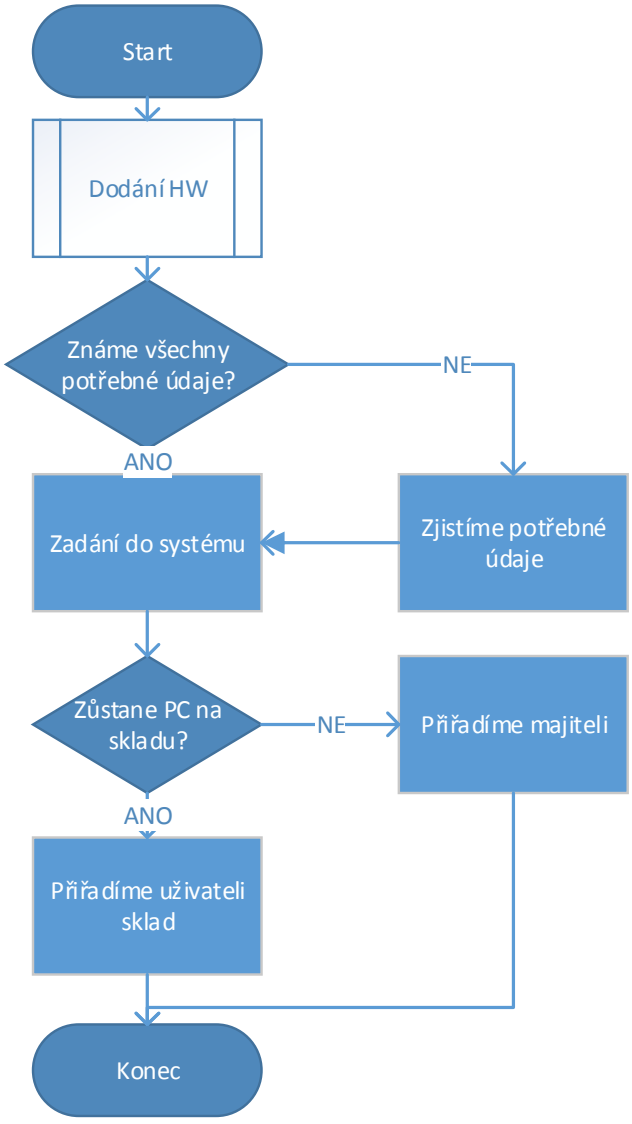
Do požadavků na databázi samozřejmě patří i konkrétní procesy, které se budou s databází provádět. Nejzásadnější z nich jsou přidání nového zařízení a vygenerování zařízení, kterým končí leasingová smlouva.

3.2.1 Změna atributu/smazání zařízení

Tento jednoduchý proces mění atributy a stav zařízení. Je nastaven tak, aby neaktivním zařízením nešly změnit atributy, tímto se vyhneme zásahů do vrácených, popř. ztracených zařízení.

3.2.2 Přidání nového zařízení

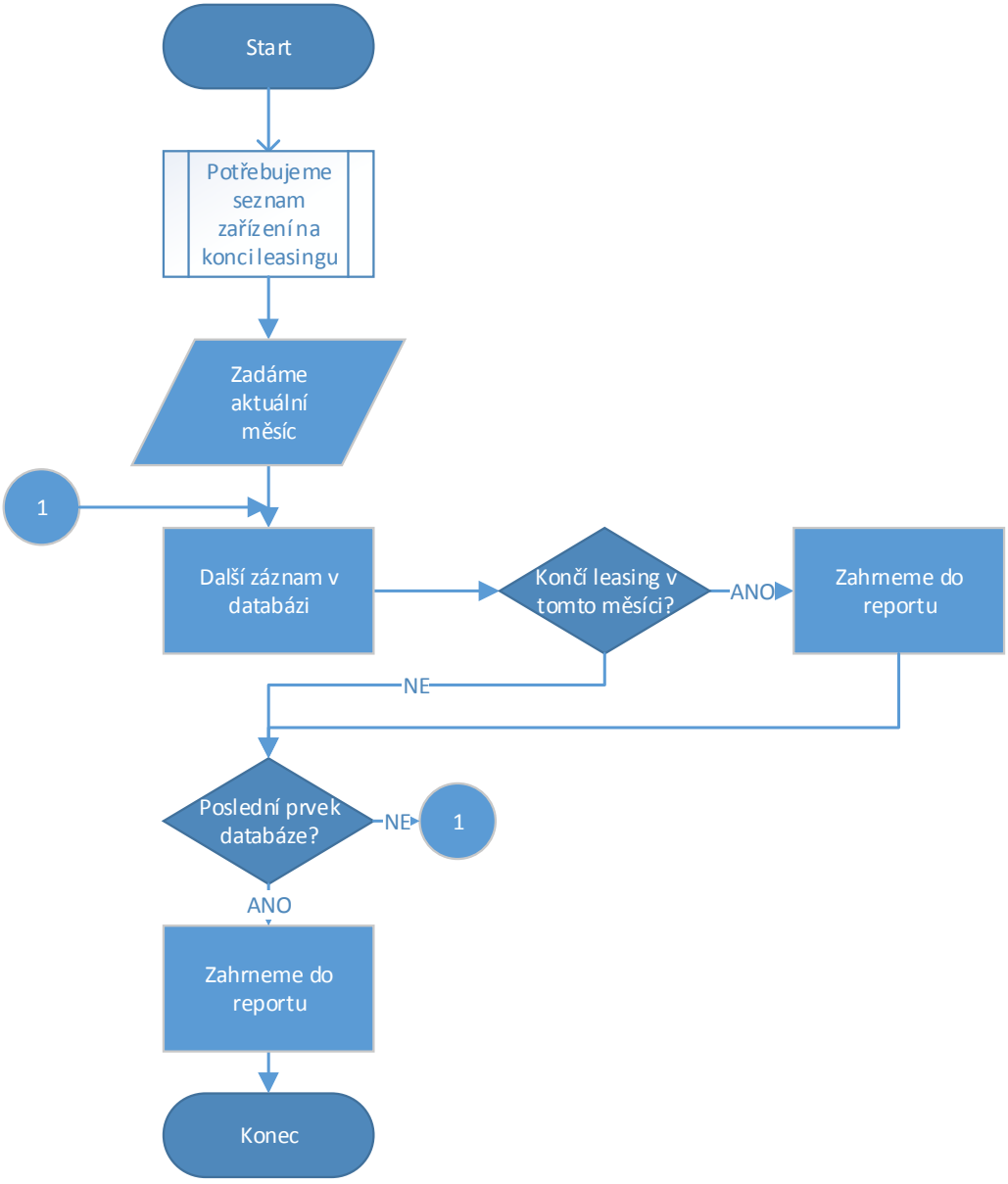
Tento proces začíná při převzetí zařízení na logistickém oddělení, poté je zaevidován do databáze a přiřazen na speciálního uživatele „sklad“, na kterého jsou přiřazena všechna zařízení, která se aktuálně nachází na skladě. Tento uživatel usnadňuje práci a šetří čas pracovníků na IT oddělení při případné inventuře, popř. dohledávání volného zařízení, díky možnosti zobrazení všech zařízení aktuálně na skladě.



Obrázek 7: Proces přidání nového zařízení (Zdroj: vlastní)

3.2.3 Vygenerování seznamu zařízení s končícím leasingem

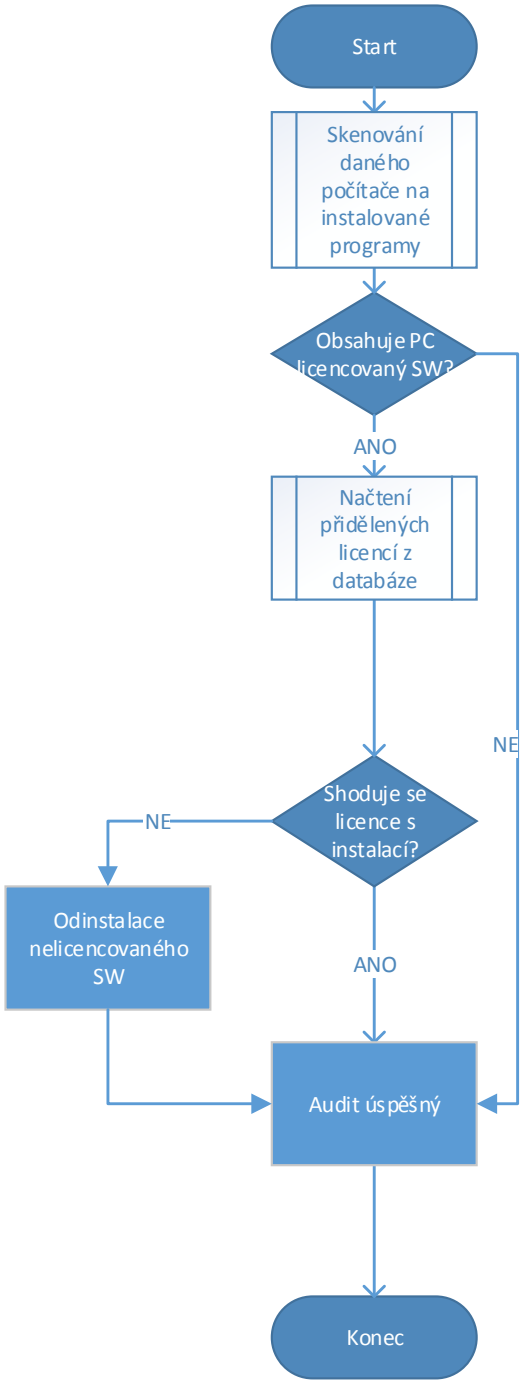
Každý měsíc je nutné vrátit zařízení vypůjčené na leasing. Databáze má vytvořený pohled na databázi, který zobrazuje všechna zařízení, kterým končí leasingová smlouva v daném měsíci, popř. exportovat tento seznam.



Obrázek 8: Vygenerování reportu (Zdroj: vlastní)

3.2.4 Audit software

Několikrát do roka probíhá ve společnosti interní audit software, který zjišťuje, nejsou-li nainstalovány programy, na které daný uživatel nemá licenci. Tyto interní audity jsou pouze preventivní a slouží k nápravě nedostatků, a zároveň šetří společnosti peníze za případné pokuty při opravdovém externím auditu.



Obrázek 9: Software audit (Zdroj: vlastní)

3.3 Konceptuální návrh

V této části jsou identifikovány entity a relace, které splňují požadavky na databázi. Z nich je ve výsledku vytvořen finální ER diagram.

3.3.1 Identifikace entit

Tabulka níže obsahuje jednotlivé entity s krátkým textovým popisem a jejich předpokládaným počtem výskytů v databázi.

Entita	Popis	Výskyt
Status	Stavy, ve kterých může zařízení být	5
Employee	Jednotliví zaměstnanci společnosti	3000
Phone	Mobilní telefony	3000
SIM_card	Sim karty	3000
SW	Programy a licence	10000
Projector	Projektory a smartboardy	150
Printer	Tiskárny a plottery	30
Monitor	Monitory	5000
typ_pc	Je-li PC laptop nebo desktop,	20
PC	Samotné PC	5000

Tabulka 1:Identifikace entit (Zdroj: vlastní)

3.3.2 Identifikace relací mezi entitami

Tabulka níže shrnuje všechny kardinality relací i s komentářem.

Entita	Kardinalita	Popis vztahu
Typ_PC - PC	1:N	PC může být pouze jednoho typu, typ může být stejný pro N počítačů
PC - Employee	1:N	Zaměstnanec může mít N počítačů, ale počítač vlastní právě jeden zaměstnanec
Status - PC	N:1	PC musí mít 1 status, ve stavu může být N počítačů

Phone - Employee	1:N	Zaměstnanec může mít N telefonů, ale telefon vlastní právě jeden zaměstnanec
SIM_card - Employee	1:N	Zaměstnanec může mít N SIM karet ale SIM kartu vlastní právě jeden zaměstnanec
SW - Employee	1:N	Zaměstnanec může mít N programů, ale program vlastní právě jeden zaměstnanec
Projector - Employee	1:N	Zaměstnanec může mít N projektorů, ale projektor vlastní právě jeden zaměstnanec
Printer - Employee	1:N	Zaměstnanec může mít N tiskáren ale tiskárnu vlastní právě jeden zaměstnanec
Monitor - Employee	1:N	Zaměstnanec může mít N monitorů, ale monitor vlastní právě jeden zaměstnanec
Status - Projector	N:1	Projektor musí mít 1 status, ve stavu může být N projektorů

Tabulka 2:Identifikace kardinalit relací (Zdroj: vlastní)

3.4 Logický návrh

Obsahuje všechny entity, jejich atributy, primární a cizí klíče a relace mezi nimi. Slouží k vytvoření tabulek, ověření integrity a normalizace.

3.4.1 Datový slovník

V datovém slovníku je každá tabulka popsána podrobně. Je to seznam všech entit a jejich atributů, typů jednotlivých atributů, délky, primárních a cizích klíčů a možných omezení. Pro tvorbu databáze je to důležitý dokument, protože se podle něj postupuje při samostatné tvorbě SQL databáze. Samotný datový slovník najdeme v příloze.
Následuje komentář vybraných entit, jejich atributů a případy jejich užití v praxi.

3.4.2 Entita Employee

Je nejdůležitější entitou této databáze, protože téměř všechny ostatní entity s ní mají nějaký vztah. Obsahuje informace o zaměstnanci jako např. identifikační číslo EID, EID přímého nadřízeného, e-mail a „cost centrum“, na které se zařízení daného uživatele

účtují. Speciálním zaměstnancem je fiktivní zaměstnanec „sklad“. Po přidání se nové zařízení, které ještě nemá majitele, převede na uživatele „sklad“. Zařízení, která se vrátí v rámci leasingu nebo jsou ztracené či ukradené, se taktéž převedou na uživatele „sklad“ avšak jejich stav se změní na „vráceno“ nebo „ztraceno“. Posledním atributem je Active, který nabývá pouze hodnot „ano“ a „ne“.

3.4.3 Entita Status

Tato entita slouží pro zaznamenávání stavu zařízení. Stavy jsou následující:

- Aktivní
- Vráceno
- Ztraceno
- Odkoupeno
- Neaktivní

3.4.4 Entita Typ PC

Tato entita slouží pro rozdělení počítačů do skupin. Typ rozlišuje počítače mezi laptopy, desktopy a tablety. Model a výrobce popisuje konkrétní číselné označení, pod kterým je počítač dostupný na trhu.

3.4.5 Entita Printer

Entita pro tiskárnu obsahuje tyto atributy:

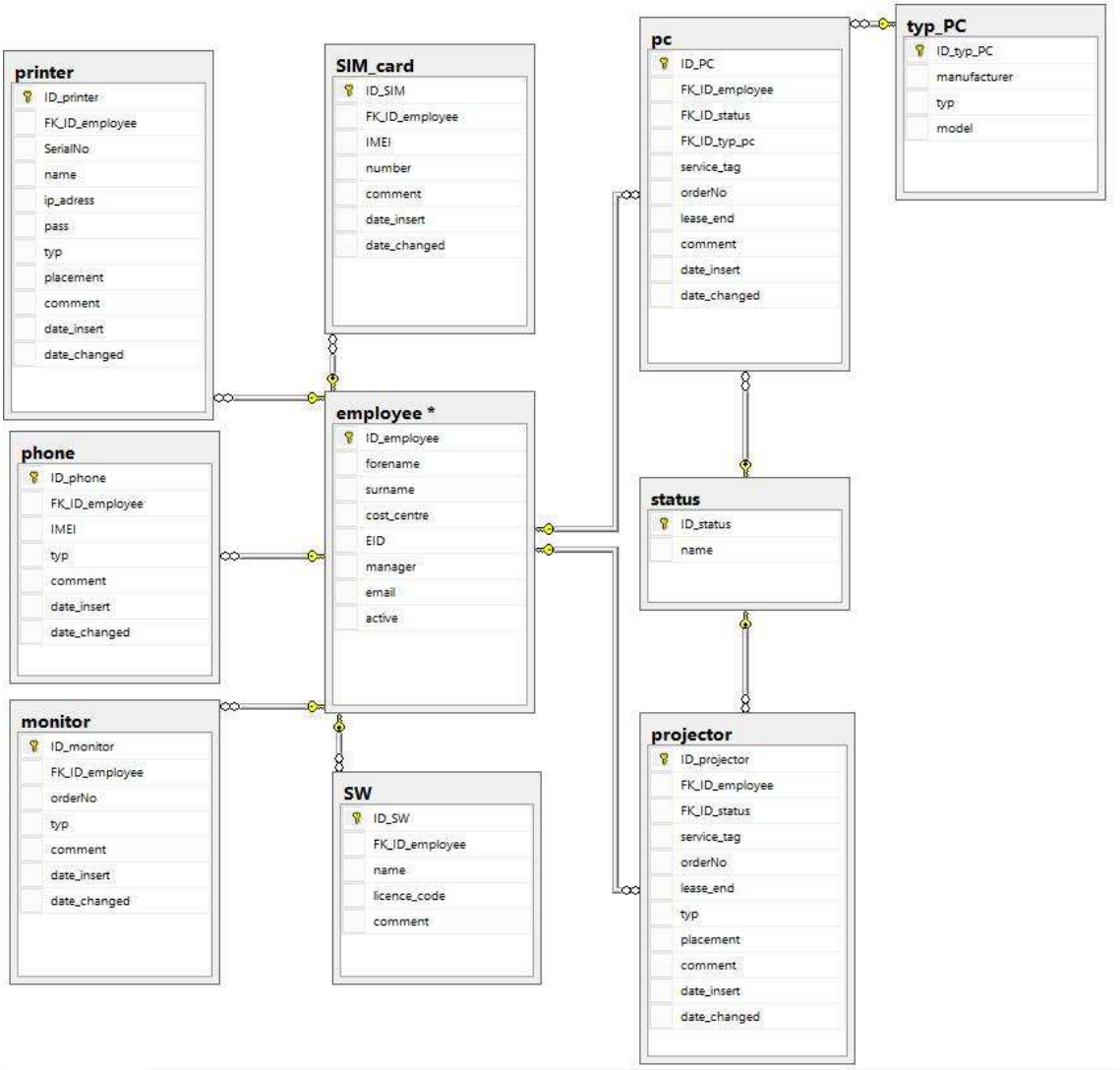
- Jméno tiskárny podle standardu společnosti CZ09PRxxx
- Sériové číslo
- IP adresu pro přístup k administrátorskému rozhraní tiskárny
- Heslo pro administrátorský účet
- Umístění
- Konkrétní typ
- Datum vložení
- Datum poslední změny

3.4.6 Entita SW

Obsahuje název SW a licenční číslo licence. Je možné do komentáře připsat poznámku s kontaktem na dodavatele, popř. podrobnosti k dané licenci.

3.4.7 ER diagram

Základním kamenem pro tvorbu databáze je ER diagram, je vymodelován entitami a relacemi uvedenými výše. Tento diagram neobsahuje žádnou vazbu M:N, a není tedy nutné dekomponovat tabulky.



Obrázek 10: ER diagram (Zdroj: vlastní)

3.5 Fyzický návrh – pohledy a procedury

Fyzický návrh se věnuje některým pohledům, procedurám a spouštím, které jsou v databázi použity. Zdrojový kód databáze včetně všech pohledů, procedur a spouští je v příloze č. 1.

3.5.1 Pohled - počítače na skladě

Často je potřeba zjistit, které počítače jsou na skladě, proto existuje tento pohled. Zobrazí název PC, jeho typ a datum, kdy se počítač musí vrátit leasingové společnosti.

```
CREATE VIEW PC_sklad AS
SELECT employee.forename, pc.service_tag AS 'Název PC', typ_PC.typ,
typ_PC.model, pc.lease_end AS 'Leasing končí'
FROM employee, pc, typ_pc
WHERE employee.ID_employee = pc.FK_ID_employee AND employee.forename =
'sklad' AND pc.FK_ID_typ_pc=typ_PC.ID_typ_PC
```

3.5.2 Pohled – monitory, které může spravovat daný uživatel

Každý manažer má možnost spravovat svá zařízení. Tento pohled slouží pro zobrazení daných zařízení z databáze.

```
CREATE VIEW uzivatel_pohled_monitory AS
SELECT employee.forename, employee.surname, monitor.ID_monitor
FROM employee, monitor
WHERE employee.ID_employee = monitor.FK_ID_employee
```

3.5.3 Procedura – všechna zařízení uživatele

Pro zjištění všech zařízení, která jsou přiřazena na konkrétního uživatele, existuje jednoduchá procedura, které po zadání EID zaměstnance vypíše všechna zařízení daného uživatele.

```
CREATE PROCEDURE vsechna_zarizeni
@EID varchar(7)
AS
BEGIN
SELECT employee.EID, pc.service_tag AS 'PC'
FROM employee, pc
WHERE employee.EID = @EID AND employee.ID_employee = pc.FK_ID_employee
SELECT employee.EID, monitor.orderNo AS 'MONITOR'
FROM employee, monitor
```

```
WHERE employee.EID = @EID AND employee.ID_employee =
monitor.FK_ID_employee
SELECT employee.EID, projector.service_tag AS 'PROJEKTOR'
FROM employee, projector
WHERE employee.EID = @EID AND employee.ID_employee =
projector.FK_ID_employee
SELECT employee.EID, printer.name AS 'PRINTER'
FROM employee, printer
WHERE employee.EID = @EID AND employee.ID_employee =
printer.FK_ID_employee
SELECT employee.EID, phone.typ AS 'PHONE'
FROM employee, phone
WHERE employee.EID = @EID AND employee.ID_employee =
phone.FK_ID_employee
SELECT employee.EID, SIM_card.number AS 'CISLO'
FROM employee, SIM_card
WHERE employee.EID = @EID AND employee.ID_employee =
SIM_card.FK_ID_employee
END
```

Příklad volání procedury:

```
EXECUTE vsechna_zarizeni 'E256496'
```

3.5.4 Procedura – přidání nového zaměstnance

Pro přidání nového zaměstnance stačí zavolat proceduru s požadovanými parametry.

```
CREATE PROCEDURE novy_zamestnanec
@jmeno VARCHAR(20),
@prijmeni VARCHAR(30),
@EID VARCHAR(7),
@costcenter VARCHAR(10),
@manager VARCHAR(7),
@mail VARCHAR(60)
AS
BEGIN
IF not exists (SELECT * FROM employee WHERE @EID = employee.EID)
INSERT INTO employee (forename, surname, EID, manager_EID,
cost_centre,email,FK_ID_status)
VALUES (@jmeno,@prijmeni,@EID, @manager, @costcenter, @mail,'1')
ELSE
PRINT ('Zaměstnanec s tímto EID už je v databázi')
END
GO
```

Příklad volání procedury:

```
EXECUTE novy_zamestnanec
'Roman','Cernik','568818','23441','E493302','roman.cernik@spolecnost.c
om'
GO
```

3.5.5 Procedura – přeřazení monitoru

Procedura slouží pro změnu majitele monitoru, obdobná procedura by šla použít pro ostatní položky databáze.

Při zadání neplatného EID uživatele, funkce hodnoty nezmění a zahlásí chybu.

```
CREATE PROCEDURE prerad_monitor
@from VARCHAR(7),
@to VARCHAR (7),
@number INT
AS
BEGIN
DECLARE @cislofrom INT;
DECLARE @cisloto INT;
SELECT @cislofrom = ID_employee FROM employee WHERE @from =
employee.eid
SELECT @cisloto = ID_employee FROM employee WHERE @to = employee.eid
IF EXISTS (SELECT * FROM monitor WHERE @cislofrom =
monitor.FK_ID_employee AND monitor.orderNo = @number)
IF EXISTS (select * from employee where @to = employee.EID)
BEGIN
UPDATE monitor
SET fk_id_employee = @cisloto
WHERE orderNo = @number
END
ELSE
PRINT ('Spatne zadani')
END
GO

EXECUTE prerad_monitor 'E837240','E000000','552334'
go
```

3.5.6 Trigger – ukončení zaměstnance

Trigger bude sloužit pro situace, kdy uživatel ukončí pracovní smlouvu a přijde vrátit všechna zařízení, která vlastní. Je navázán na akci smazání uživatele. Místo smazání uživatele proběhne funkce, která změní stav uživatele na neaktivní a provede kontrolu všech zařízení, která jsou pod uživatelem vedená, převede je na sklad.

```
CREATE TRIGGER vystup ON employee
INSTEAD OF DELETE
AS
DECLARE @emp_id INT;
SELECT @emp_id = d.ID_employee FROM deleted d
BEGIN
	UPDATE employee
	SET active = 0
	WHERE employee.ID_employee = @emp_id
```

```
        UPDATE monitor
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
        UPDATE pc
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
        UPDATE phone
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
        UPDATE printer
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
        UPDATE projector
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
        UPDATE SIM_card
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
        UPDATE SW
        SET FK_ID_employee = '1'
        WHERE FK_ID_employee = @emp_id
END
GO

DELETE employee WHERE employee.eid = 'E837240'
```

3.6 Možná rozšíření databáze

Databáze může sloužit jako přehledný datový podklad pro další systémy. Tyto systémy mohou zjednodušit komunikaci a fungování společnosti. Následuje několik systémů a rozšíření, které by mohly tuto konkrétní databázi využívat.

3.6.1 Systém pro objednávání

Základem tohoto systému by byl jednoduchý formulář. Do něj by se zadalo EID uživatele, pomocí kterého by se doplnila hlavička objednávky. Po výběru, zda chce uživatel objednat telefon, počítač, monitor, telefon nebo licenci programu, by se zobrazily možné typy daného zařízení z databáze. Po zvolení typu by se celá objednávka poslala e-mailem přímému nadřízenému ke schválení.

3.6.2 Půjčovna

Systém pro půjčování zařízení by zobrazoval pouze zařízení s atributem půjčovací. Tato zařízení by byla připravena pro případy, kdy je potřeba zajistit náhradní telefon nebo počítač, kdy uživatel se stolním počítačem musí náhle odjet na služební cestu,

popř. při opravě svého zařízení. Dále by systém musel obsahovat kalendář, pomocí kterého by daná zařízení bylo možné zarezervovat.

3.7 Celkové hodnocení

Aktuálně existují tři různé databáze, které v navrhovaném řešení tvoří jednu databázi obsahující všechna potřebná data pro lokální účely. Procesy zůstávají zachovány. Proces pro výměnu počítačů v leasingu se v navrhovaném řešení podařilo zlepšit díky pohledu na databázi, která zobrazuje všechny počítače, které je potřeba v daném měsíci vrátit. Data z účetního exportu se nemusí párovat manuálně po jednom zařízení, jako v současném stavu, ale stačí zkontrolovat, jestli sedí dva exportované soubory.

Při nástupu nového člověka do společnosti je aktuálně nutné přiřadit všechna jeho zařízení v systému. Podařilo se snížit počet přístupů ze tří databází, na databázi jednu. Počet jednotlivých kroků nebylo nutné snižovat, protože výsledkem je správa zařízení pouze v jedné databázi, která je pro uživatele i administrátora jednodušší.

Kdykoliv odchází uživatel ze společnosti, je nutné jej ve všech třech databázích zrušit a jeho zařízení převést na sklad nebo jeho nadřízeného. Tuto funkci v nové databázi řeší trigger, který místo zrušení uživatele, změní stav účtu na neaktivní a všechna zařízení automaticky převede na sklad. Tento trigger sníží délku procesu výstup uživatele (kapitola 2.4) ze šesti na dva kroky, a to kontrolu všech zařízení uživatele a vrácení všech zařízení.

ZÁVĚR

Cílem bakalářské práce bylo navrhnout databázi, která by zjednodušila práci zaměstnanců IT oddělení, při správě zařízení a programů ve společnosti. Zaměřil jsem se na nejčastější procesy, které by se s databází měly provádět a jejich implementací do databáze.

Databáze je navržena tak, aby neobsahovala nic navíc, než je třeba pro každodenní správu zařízení na IT oddělní. Globální databáze nejsou pro tuto každodenní rutinní práci vhodné, kvůli své složitosti a rozsahu. Na každé pobočce společnosti se interní procesy lehce liší a nelze proto při návrhu počítat se všemi možnými variantami.

Globálním řešením by mohlo být vytvoření korporátní šablony pro databázi, kterou by si jednotlivá IT oddělení na pobočkách mohla upravovat podle svých potřeb.
Pro usnadnění a zvýšení komfortu práce s databází by určitě pomohlo jednoduché uživatelské prostředí, do kterého by se přistupovalo přes webový prohlížeč. Pomocí záložek by byla vytvořena nabídka přednastavených pohledů. Dále by bylo možné zasahovat do databáze pomocí tlačítek, které by spouštěly připravené procedury.

SEZNAM POUŽITÉ LITERATURY

(1) CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Vyd. 1. Brno: Computer Press, 2009, 584 s. ISBN 978-80-251-2328-7.

(2) KOCH, Miloš, Carolyn E BEGG a Richard HOLOWCZAK. *Datové a funkční modelování: profesionální průvodce tvorbou efektivních databází*. Vyd. 3., přeprac. Brno: Akademické nakladatelství CERM, 2008, 121 s. ISBN 978-80-214-3731-9.

(3) KŘÍŽ, Jiří, Carolyn E BEGG a Richard HOLOWCZAK. *Databázové systémy: profesionální průvodce tvorbou efektivních databází*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2005, 111 s. ISBN 80-214-3064-8.

(4) GILFILLAN, Ian. Myslíme v jazyce MySQL 4. Vyd. 1. Praha: Grada, 2003, 750 s. ISBN 80-247-0661-X.

(5) HROMEK, Jiří. Databázové systémy [online]. Olomouc, 2007 [cit. 2015-05-19]. Dostupné také z: http://phoenix.inf.upol.cz/esf/ucebni/databa.pdf.

(6) CORONEL, Carlos, Steven MORRIS a Peter ROB. Database systems: design, implementation, and management. 10e [edition]. Boston, MA: Course Technology/Cengage Learning, 2013, xxvi, 726 s.

(7) HOULETTE, Forrest. SQL: příručka programátora. Praha: SoftPress, 2001, 382 s. ISBN 80-864-9714-3.

(8) Sas 9.4 sql procedure user's guide. S.l.: Sas, 2013. ISBN 978-161-2905-686.

(9) WATT, Andrew. Microsoft SQL Server 2005 programming for dummies. Hoboken, N.J.: Wiley, 2007, xvi, 416 s. --For dummies. ISBN 978-047-1774-228.

(10) SKLENÁK, Vilém. Data, informace, znalosti a Internet. Vyd. 1. V Praze: C.H. Beck, 2001, xvii, 507 s. C.H. Beck pro praxi. ISBN 80-717-9409-0.

(11) Databázové modely. Databáze [online]. Copyright©2010 [cit. 2016-01-18]. Dostupné z: http://www.databaze.chytrak.cz/modely.htm

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

Příloha č. 1: Zdrojový kód

```
CREATE TABLE status
(
	ID_status int identity(1,1) primary key not null,
	name varchar(30)
)
go

CREATE TABLE employee
(
	ID_employee int identity(1,1) primary key not null,
	forename varchar(20),
	surname varchar(30),
	cost_centre varchar(15),
	EID varchar(7) not null,
	manager varchar(7),
	email varchar(50),
	active binary
	)
GO

CREATE TABLE phone
(
	ID_phone int identity(1,1) primary key not null,
	FK_ID_employee int not null,
	IMEI int not null,
	typ varchar(40),
	comment varchar(100),
	date_insert date,
	date_changed date,
	CONSTRAINT FK_ID_employee_phone FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee)
)
GO

CREATE TABLE SIM_card
(
	ID_SIM int identity(1,1) primary key not null,
	FK_ID_employee int not null,
	IMEI int not null,
	number int not null,
	comment varchar(100),
	date_insert date,
	date_changed date,
	CONSTRAINT FK_ID_employee_sim FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee)
)
GO

CREATE TABLE SW
(
```

```
	ID_SW int identity(1,1) primary key not null,
	FK_ID_employee int not null,
	name varchar(30) not null,
	licence_code varchar(30),
	comment varchar(100),
	CONSTRAINT FK_ID_employee_sw FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee)
)
GO

CREATE TABLE printer
(
	ID_printer int identity(1,1) primary key not null,
	FK_ID_employee int not null,
	SerialNo int not null,
	name varchar(10) not null,
	ip_adress varchar(15),
	pass varchar(15),
	typ varchar(30),
	placement varchar(30),
	comment varchar(100),
	date_insert date,
	date_changed date,
	CONSTRAINT FK_ID_employee_printer FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee)
)
GO

CREATE TABLE projector
(
	ID_projector int identity(1,1) primary key not null,
	FK_ID_employee int not null,
	FK_ID_status int not null,
	service_tag varchar(7) not null,
	orderNo int not null,
	lease_end date,
	typ varchar(30),
	placement varchar(30),
	comment varchar(100),
	date_insert date,
	date_changed date,
	CONSTRAINT FK_ID_employee_projector FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee),
	CONSTRAINT FK_ID_status_projector FOREIGN KEY (FK_ID_status) REFERENCES
status(ID_status),

)
GO

CREATE TABLE monitor
(
	ID_monitor int identity(1,1) primary key not null,
```

```
	FK_ID_employee int not null,
	orderNo int not null,
	typ varchar(30),
	comment varchar(100),
	date_insert date,
	date_changed date,
	CONSTRAINT FK_ID_employee_monitor FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee)
)
GO


CREATE TABLE typ_PC
(
	ID_typ_PC int identity(1,1) primary key not null,
	manufacturer varchar(30),
	typ varchar(20),
	model varchar(10)

)
GO

CREATE TABLE pc
(
	ID_PC int identity(1,1) primary key not null,
	FK_ID_employee int not null,
	FK_ID_status int not null,
	FK_ID_typ_pc int not null,
	service_tag varchar(7) not null,
	orderNo int not null,
	lease_end date,
	comment varchar(100),
	date_insert date,
	date_changed date,
	CONSTRAINT FK_ID_employee_PC FOREIGN KEY (FK_ID_employee) REFERENCES
employee(ID_employee),
	CONSTRAINT FK_ID_status_PC FOREIGN KEY (FK_ID_status) REFERENCES
status(ID_status),
	CONSTRAINT FK_ID_typ_pc_PC FOREIGN KEY (FK_ID_typ_PC) REFERENCES
typ_pc(ID_typ_pc),

)
GO

--pridani stavu
INSERT INTO status (name) VALUES ('aktivní')
INSERT INTO status (name) VALUES ('vráceno')
INSERT INTO status (name) VALUES ('ztraceno')
INSERT INTO status (name) VALUES ('odkoupeno')
INSERT INTO status (name) VALUES ('neaktivní')
GO
```

```
--pridani uzivatelu
INSERT INTO employee (forename,surname,EID,active) VALUES
('sklad','sklad','E000000',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Ondrej','Vymola','E837240',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Roman','Zicha','E456332',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Radek','Poledna','E123444',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Michal','Bajsikl','E223585',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Lukas','Janca','E366985',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Martin','Leska','E489362',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Leos','Kares','E217999',1)
INSERT INTO employee (forename,surname,EID,active) VALUES
('Lubomir','Vymola','E155889',1)
GO

--pridani telefonu
INSERT INTO phone (FK_ID_employee,IMEI,typ) VALUES ('1','24556688','nokia 132')
INSERT INTO phone (FK_ID_employee,IMEI,typ) VALUES ('2','54556689','iPhone 6')
INSERT INTO phone (FK_ID_employee,IMEI,typ) VALUES ('3','24556690','nokia 132')
INSERT INTO phone (FK_ID_employee,IMEI,typ) VALUES ('1','24556691','nokia 132')
GO

--pridani SIM kart
INSERT INTO SIM_card (FK_ID_employee,IMEI,number) VALUES
('1','334442','761556442')
INSERT INTO SIM_card (FK_ID_employee,IMEI,number) VALUES
('2','334443','761556443')
INSERT INTO SIM_card (FK_ID_employee,IMEI,number) VALUES
('3','334444','761556444')
INSERT INTO SIM_card (FK_ID_employee,IMEI,number) VALUES
('1','334445','761556445')
GO

--pridani SW
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('2','Adobe Acrobat
11','ADO-011-CCA')
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('3','Adobe Acrobat
11','ADO-011-CCB')
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('5','Adobe Acrobat
11','ADO-011-CCC')
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('1','Adobe Acrobat
11','ADO-011-CCD')
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('1','Adobe Acrobat
11','ADO-011-CCE')
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('2','Microsoft Project
2007','PRO-007-001')
```

```
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('2','Microsoft Project 2010','PRO-010-001')
INSERT INTO SW (FK_ID_employee,name,licence_code) VALUES ('4','Mindmap','MIN-000-896')
GO

-- pridani projektoru
INSERT INTO projector (FK_ID_employee,FK_ID_status,service_tag,orderNo) VALUES ('1','1','PRO0001','889632')
INSERT INTO projector (FK_ID_employee,FK_ID_status,service_tag,orderNo) VALUES ('4','1','PRO0002','889655')
INSERT INTO projector (FK_ID_employee,FK_ID_status,service_tag,orderNo) VALUES ('1','1','PRO0003','889669')
GO

-- pridani tiskaren
INSERT INTO printer (FK_ID_employee,SerialNo,name) VALUES ('6','255633','CZ09PR001')
INSERT INTO printer (FK_ID_employee,SerialNo,name) VALUES ('6','255634','CZ09PR002')
INSERT INTO printer (FK_ID_employee,SerialNo,name) VALUES ('6','255635','CZ09PR003')
GO

-- pridani monitoru
INSERT INTO Monitor (FK_ID_employee,OrderNo) VALUES ('2','552334')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('2','552335')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('3','552336')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('3','552337')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('4','552338')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('4','552339')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('5','552340')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('5','552341')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('6','552342')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('6','552343')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('7','552344')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('7','552345')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('8','552346')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('8','552347')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('9','552348')
INSERT INTO monitor (FK_ID_employee,OrderNo) VALUES ('9','552349')
GO

-- pridani typu PC
INSERT INTO typ_PC (manufacturer,typ,model) VALUES ('Dell','laptop','M4800')
INSERT INTO typ_PC (manufacturer,typ,model) VALUES ('Dell','laptop','E5440')
INSERT INTO typ_PC (manufacturer,typ,model) VALUES ('Dell','laptop','E7440')
INSERT INTO typ_PC (manufacturer,typ,model) VALUES ('Dell','laptop','E7240')
INSERT INTO typ_PC (manufacturer,typ,model) VALUES ('Dell','desktop','3020')
INSERT INTO typ_PC (manufacturer,typ,model) VALUES ('Dell','desktop','T5510')
GO
```

```
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('2','1','1','PCLT001','63556','2015-05-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_status,FK_ID_typ_pc,service_tag,orderNo,lease_end)
            VALUES ('3','2','1','PCLT002','63557','2015-07-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('4','2','1','PCLT003','63558','2016-05-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('5','3','1','PCLT004','63559','2015-05-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('6','4','1','PCLT005','63560','2017-05-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('7','5','1','PCDT001','63561','2016-05-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('8','1','1','PCLT006','63562','2015-08-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('9','1','1','PCLT007','63563','2017-01-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('1','5','1','PCDT002','63564','2015-05-31')
INSERT                                  INTO                                    PC
(FK_ID_employee,FK_ID_typ_pc,FK_ID_status,service_tag,orderNo,lease_end)
            VALUES ('1','6','1','PCDT003','63565','2016-05-31')

GO

CREATE VIEW PC_sklad AS
SELECT employee.forename, pc.service_tag AS 'Název PC', typ_PC.typ, typ_PC.model,
pc.lease_end AS 'Leasing konèí'
FROM employee, pc, typ_pc
WHERE employee.ID_employee = pc.FK_ID_employee AND employee.forename = 'sklad' AND
pc.FK_ID_typ_pc=typ_PC.ID_typ_PC
GO

CREATE PROCEDURE vsechna_zarizeni
@EID varchar(7)

AS
BEGIN
SELECT employee.EID, pc.service_tag AS 'PC' FROM employee, pc WHERE employee.EID =
@EID AND employee.ID_employee = pc.FK_ID_employee
SELECT  employee.EID,  monitor.orderNo  AS  'MONITOR'  FROM  employee,  monitor  WHERE
employee.EID = @EID AND employee.ID_employee = monitor.FK_ID_employee
SELECT employee.EID, projector.service_tag AS 'PROJEKTOR' FROM employee, projector
WHERE employee.EID = @EID AND employee.ID_employee = projector.FK_ID_employee
```

```
SELECT employee.EID, printer.name AS 'PRINTER' FROM employee, printer WHERE
employee.EID = @EID AND employee.ID_employee = printer.FK_ID_employee
SELECT employee.EID, phone.typ AS 'PHONE' FROM employee, phone WHERE employee.EID
= @EID AND employee.ID_employee = phone.FK_ID_employee
SELECT employee.EID, SIM_card.number AS 'CISLO' FROM employee, SIM_card WHERE
employee.EID = @EID AND employee.ID_employee = SIM_card.FK_ID_employee
END

GO

CREATE PROCEDURE novy_zamestnanec
@jmeno VARCHAR(20),
@prijmeni VARCHAR(30),
@EID VARCHAR(7),
@costcenter VARCHAR(10),
@manager VARCHAR(7),
@mail VARCHAR(60)
AS
BEGIN
IF not exists (SELECT * FROM employee WHERE @EID = employee.EID)
INSERT INTO employee (forename, surname, EID, manager, cost_centre,email,active)
VALUES (@jmeno,@prijmeni,@EID, @manager, @costcenter, @mail,1)
ELSE
PRINT ('Zaměstnanec s tímto EID už je v databázi')
END
GO

CREATE PROCEDURE prerad_monitor
@from varchar(7),
@to varchar(7),
@number int
AS
BEGIN
DECLARE @cislofrom int;
DECLARE @cisloto int;
select @cislofrom = ID_employee from employee where @from = employee.eid
select @cisloto = ID_employee from employee where @to = employee.eid
IF exists (select * from monitor where @cislofrom = monitor.FK_ID_employee AND
monitor.orderNo = @number)
IF exists (select * from employee where @to = employee.EID)
BEGIN
UPDATE monitor
SET fk_id_employee = @cisloto
WHERE orderNo = @number
END
ELSE
PRINT ('Spatne zadani')
END
GO

execute prerad_monitor 'E837240','E000000','552334'
go
```

```
create trigger vystup on employee
instead of delete
as
declare @emp_id int;
select @emp_id = d.ID_employee from deleted d
BEGIN
	update employee
	set active = 0
	where employee.ID_employee = @emp_id

	update monitor
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

	update pc
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

	update phone
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

	update printer
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

	update projector
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

	update SIM_card
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

	update SW
	set FK_ID_employee = '1'
	where FK_ID_employee = @emp_id

END
GO
select * from projector
delete employee where employee.eid = 'E837240'
select * from projector
```

Příloha č. 2: Datový slovník databáze

Entita	Atribut	Typ	Délka	Klíč	Omezení
Status	ID_status	int		PK	not null
	name	varchar	30		not null
Employee	ID_employee	int		PK	not null
	Forename	varchar	20		
	Surname	varchar	30		
	Cost_centre	varchar	15		
	EID	varchar	7		not null
	Manager_EID	varchar	7		
	Active	Boolean			not null
	Email	typ	50		
Phone	ID_phone	int		PK	not null
	FK_ID_employee	int		FK	not null
	IMEI	int			not null
	typ	varchar	40		
	Comment	varchar	100		
	Date_insert	date			
	Date_changed	date			
SIM_card	ID_SIM	int		PK	not null
	FK_ID_employee	int		FK	not null
	IMEI	int			not null
	Number	int			not null
	PUK1	varchar	10		
	Comment	varchar	100		
	Date_insert	date			
	Date_changed	date			
SW	ID_SW	int		PK	not null
	FK_ID_employee	int		FK	not null
	Name	varchar	30		not null

	LicenceCode	varchar	30		
	Comment	varchar	100		
Projector	ID_projector	int		PK	not null
	FK_ID_employee	int		FK	not null
	FK_ID_status	int		FK	not null
	ServiceTag	varchar	7		not null
	OrderNo	int			not null
	LeaseEnd	date			
	typ	varchar	30		
	Placement	varchar	30		
	Comment	varchar	100		
	Date_insert	date			
	Date_changed	date			
Printer	ID_Printer	int		PK	not null
	FK_ID_employee	int		FK	not null
	SerialNo	int			not null
	Name	varchar	10		not null
	IP_adress	varchar	15		
	Pass	varchar	20		
	typ	varchar	30		
	Placement	varchar	30		
	Comment	varchar	100		
	Date_insert	date			
	Date_changed	date			
Monitor	ID_monitor	int		PK	not null
	FK_ID_employee	int		FK	not null
	OrderNo	int			not null
	typ	varchar	30		
	Comment	varchar	100		
	Date_insert	date			
	Date_changed	date			

typ_PC	ID_typ_PC	int		PK	not null
	Manufacturer	varchar	30		
	typ	varchar	20		
	Model	varchar	10		
PC	ID_PC	int		PK	not null
	FK_ID_typ_PC	int		FK	not null
	FK_ID_employee	int		FK	not null
	FK_ID_status	int		FK	not null
	ServiceTag	varchar	7		not null
	OrderNo	int			not null
	LeaseEnd	date			
	Comment	varchar	100		
	Date_insert	date			
	Date_changed	date			