VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

ANALÝZA SQL DATABÁZE A NÁVRH NA JEJÍ ZLEPŠENÍ VE SPOLEČNOSTI

ANALYSIS OF SQL DATABASE AND PROPOSAL FOR ITS IMPROVEMENT IN THE COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jan Josífek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jiří Kříž, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Jan Josífek**
Vedoucí práce:	**Ing. Jiří Kříž, Ph.D.**
Akademický rok:	2021/22
Studijní program:	Manažerská informatika

Garant studijního programu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Analýza SQL databáze a návrh na její zlepšení ve společnosti

Charakteristika problematiky úkolu:

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

Cíle, kterých má být dosaženo:

Cílem práce je optimalizace databáze a návrh na její efektivnější provoz.

Základní literární prameny:

BEGG, C., R. HOLOWCZAK a T. CONOLLY. Mistrovství - Databáze : Profesionální průvodce tvorbou efektivních databází. Praha: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.

KOCH, Miloš a Viktor ONDRÁK. Informační systémy a technologie. Vyd. 3. Brno: Akademické nakladatelství CERM, 2008, 166 s. : il., grafy, tab. ISBN 978-80-214-3732-6.

KOCH, Miloš a Bernard NEUWIRTH. Datové a funkční modelování. Vyd. 4., rozšířené. Brno: Akademické nakladatelství CERM, 2010, 142 s. : il., grafy, tab. ISBN 978-80-214-4125-5.

STEPHENS, R.,R. PLEW a A. D.JONES. Naučte se SQL za 28 dní. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2021/22

V Brně dne 28.2.2022

L. S.

Ing. Jiří Kříž, Ph.D.
garant

doc. Ing. Vojtěch Bartoš, Ph.D.
děkan

Abstrakt

Tato bakalářská práce se zabývá analýzou a návrhem zlepšení databáze ve společnosti. Práce je rozdělena na tři části. V první části jsou popsána teoretická východiska, která slouží jako podklad pro další části. V druhé části je analýza současného stavu databáze a ve třetí části je navrženo vlastní řešení problému.

Klíčová slova

Data, Databáze, SQL, normalizace, Datový model, ER diagram

Abstract

This bachelor thesis deals with the analysis and design of database improvement in the company. The thesis is divided into three parts. In the first part, the theoretical background is described, which serves as a basis for the other parts. The second part analyses the current state of the database and the third part proposes the actual solution to the problem.

Key words

Data, Database, SQL, normalization, Data model, ER diagram

Bibliografická citace

JOSÍFEK, Jan. Analýza SQL databáze a návrh na její zlepšení ve společnosti. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/143755. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Ing. Jiří Kříž Ph.D.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracovala jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušila autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně, dne 7.5.2022

……

podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu práce, panu Ing. Jiřímu Křížovi, Ph.D. za vedení bakalářské práce a jeho ochotu a trpělivost. A mé rodině za psychickou a finanční podporu.

Obsah

Úvod

Počátky databází sahají hluboko do historie, když člověk začal poprvé psát data na kamenné desky a později na papír. Ve starověku byly vyvinuty složitější databázové systémy, které umožňovali ukládat informace na jediném místě například knihovny, nemocnice, obchodní organizace. Od té doby až po současnost si databáze a databázové systémy prošly velkým vývojem hlavně od vzniku prvních počítačů a díky tomu vznikly elektronické databáze a databázové systémy.

V dnešní době je databáze jedním z nejzákladnějších prvků v informačních technologiích využívaných ve světě, ať už jde o státní orgány, společnosti, školy, zdravotnictví apod. Díky výkonu dnešní výpočetní techniky je ukládání a práce s daty snadnější a rychlejší.

Bakalářská práce se zaměřuje na analýzu současného stavu databáze ve vybrané společnosti a návrhem pro zlepšení současného stavu databáze.

Cíl práce, metody a postupy zpracování

Cílem práce je optimalizace databáze a návrh na její efektivnější provoz. Práce obsahuje tři části. První částí jsou teoretická východiska, která popíšou základní pojmy o tématu práce a slouží jako podklad pro zbývající části práce. Druhou částí je seznámení se s podnikem, popis informačních technologií a analýza současného stavu databáze z hlediska zabezpečení, definice tabulek a relací, normalizace. V poslední třetí části je popsán návrh řešení na základě analýzy pro odstranění nedostatků databáze.

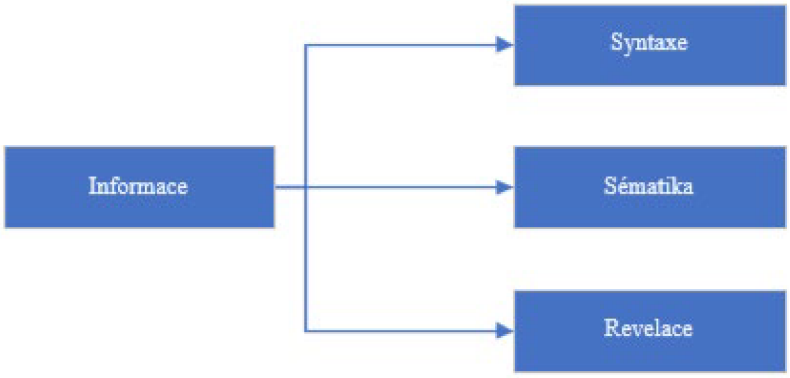
1 Teoretická východiska práce

1.1 Informace, data a znalosti

1.1.1 Informace

Informace má hodně významů a lze se na ně dívat z více hledisek. Obecně můžeme definovat informaci jako zprávu nebo údaj, která splňuje 3 požadavky:

- Syntetická revelace – detekovat a porozumět jí.
- Sémantická relevance – pochopit co znamená.
- Pragmatická revelace – musí mít nějaký význam pro příjemce.

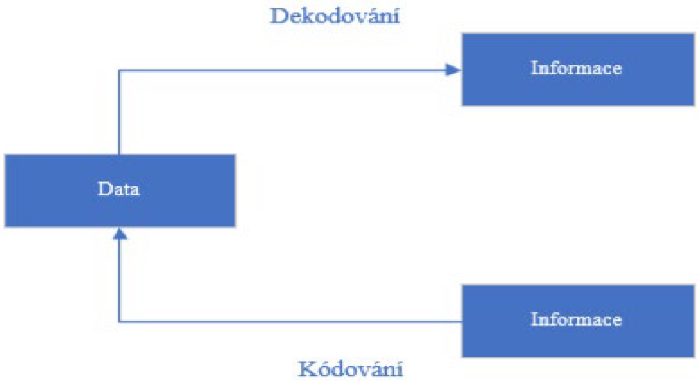


Obrázek č. 1: informace (zdroj: [1] s.4)

Informace můžeme rozdělit do skupin podle různých stanovisek například: operativní, strategické, taktické, krátkodobé, dlouhodobé a další. [1]

1.1.2 Data

Data se stávají informací, pokud se používají k rozhodování a mají určitý význam pro uživatele. Každý je ve styku s informacemi (zprávami), a pokud je zachytí a porozumí jim, stanou se pro člověka daty. Tato data může člověk přeměnit a zaznamenat na jakýkoliv fyzický nosič, ať už je to papír, počítač nebo elektronické signály. Tento proces se nazývá kódování (informace se stávají daty) a pro příjemce se dekódují (data se převedou na informaci). [1]



Obrázek č. 2: Data (zdroj: [1] s.5)

1.1.3 Znalosti

Znalosti jsou informace o tom, jak lze použít další informace a data. Lidé získávají znalosti zkušenostmi, praxí nebo studiem a znalost každého člověka je unikátní. Rozhodujeme se na základě znalostí. [1]

1.2 Databázový systém (DS)

Důvodem k vytvoření databázového systému bylo odstranění problémů se zpracováváním velkých objemů dat. Před vznikem databázových systémů aplikace prováděly komunikaci s uživateli, vlastní výpočty, práce s daty, a to způsobovalo problémy, jako byla nekompatibilita dat a problémy s připojením více uživatelů k datům.

DS shromažďuje informace, zpracovává a ukládá je na jednom centrálním místě. Díky tomu se efektivně pracuje s informacemi (rychlé vyhledávání, změny stávajících údajů, matematické operace, pohledy a další operace).

Databázový systém se skládá ze systému řízení báze dat (DBMS), databáze a aplikace. [2, 3]

1.2.1 Systém řízení báze dat (DBMS)

Je to softwarový systém, který organizuje a skladuje informace. DBSM poskytuje tyto služby:

- Definice dat – umožňuje definovat a uchovat datové entity.

- Údržba dat – stará se o entitu, vyčleňuje jejím členům záznamy, které se skládají z položek popisujících dílčí informace o členu.
- Manipulace s daty – umožňuje uživateli vkládat, aktualizovat, rušit a třídit data v databázi.
- Zobrazování údajů – poskytuje prezentaci dat uživateli.
- Integrita dat – zajišťuje správnost dat.

Další důležité vlastnosti jsou podpora transakcí a řízení zotavení. Transakce je posloupnost několik akcí, které pracují s daty v databázi a probíhají jako celek. Kvůli složitosti takové akce může docházet k nekonzistenci dat, a proto se DBMS stará o to, aby se provedly všechny změny nebo žádná. V případě že by došlo k nekonzistenci dat, ať už kvůli transakcí nebo selhání softwaru nebo hardwaru, DBMS poskytuje mechanismy k navrácení dat do původního stavu. [2, 4]

1.2.2 Databáze

Databázi si lze představit jako soubor dat, které jsou uložené podle určitých pravidel. Před vznikem počítačů a databázových aplikací byla typická databáze knihovna nebo kartotéka, kde jsou záznamy uložené podle pravidel. Databáze obsahuje datové prvky, vztahy, integritní omezení a schéma. [2]

1.2.2.1 Rozdělení databází z pohledu jejich řízení

a) Transakční databáze (OLTP) - data jsou aktuální a neustále se mění. Hlavním úkolem je sbírání a správa dat v podnicích. Evidují se zde každodenní operace jako jsou prodeje, inventarizace, bankovnictví, výroba, výplaty atd.
b) Analytické databáze (OLAP) – data jsou historická a nemění se. Ve většině případů se extrahují z transakčních databází. OLAP se využívá pro zobrazení statistických dat, trendů a podobně. [5]

Tabulka č. 1: rozdíl mezi OLTP a OLAP databází (Zdroj: Vlastní zpracování dle: [5])

Odlišné vlastnosti	OLTP	OLAP
Uživatelé	Zákazník	Obchodník
Datový obsah	Současná, detailní	Historická, sloučená
Návrh databáze	ER model + aplikace	Schéma hvězdy + subjekt
Pohled na data	Aktuální, lokální	Evoluční, integrovaný
Přístupové vzory	Aktualizace	Read-only, komplexní dotazy

1.2.2.2 Entita

Data jsou pro nás určitá entita informací, která nás zajímají. V první řadě jdou definovat a mají pro nás význam. Entitou mohou být žáci a každý záznam obsahuje informace o jednom členu entity. [1]

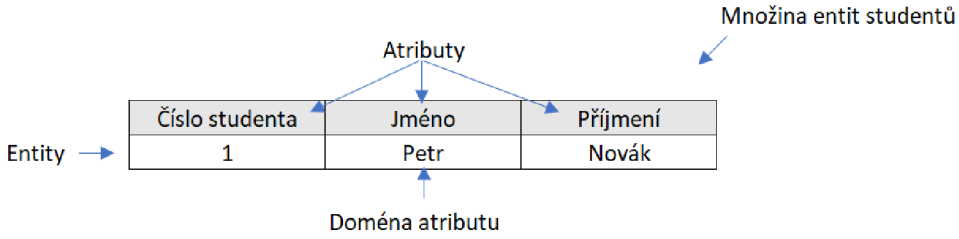
1.2.2.3 Atribut

Atribut neboli datová položka je dále nedělitelný údaj entity, který chceme uchovávat. U entity žáka se může jednat například o atributy číslo studenta, jméno a příjmení. [1]

1.2.2.4 Doména atributu

Je to hodnota atributu například Petr u atributu Jméno. [1]

1.2.2.5 Schéma tabulky



Obrázek č. 3: Schéma tabulky z teorie množin (Zdroj: [1] s. 24)

1.2.2.6 Datový typ a délka

Atribut může mít definovanou pevnou délku (pěvně stanoveno kolik bude mít položka znaků) nebo proměnnou délku (položka bude mít délku takovou, kolik do ní vložíme znaků). Dále se deklaruje datový typ. Základní datové typy jsou text, čísla a datum. [1]

1.2.2.7 Metadata

Je to popis uložených dat v databázi (data o datech). Například odkud data pochází, datový typ, délka atd. Tato data následně tvoří v databázích systémové katalogy nebo slovník dat. [1]

1.2.3 Databázové modely

1.2.3.1 Hierarchický model

Hierarchie se tvoří mezi entitami se vztahy nadřízeností a podřízeností a data jsou uspořádaná do stromové struktury. [2, 6]

1.2.3.2 Síťový model

Modeluje se v grafech. Uzly jsou entity a orientované hrany jsou vztahy mezi entitami. [2, 6]

1.2.3.3 Souborově orientovaná databáze

Používá ISAM metodu (indexsekvenční metodu). Pro každou tabulku se vytváří samostatný soubor. [2, 6]

1.2.3.4 Relační model

V dnešní době je nejpoužívanější model relační. Již z názvu vyplývá, že je založený na relacích. Relaci si můžeme představit jako tabulku, která se skládá ze sloupců a řádků. Atributy entit jsou obsažené ve sloupcích a údaje v řádku obsahují aktuální stav. Relace mají mezi sebou relační vztah. [1, 2]

1.2.3.5 Objektově orientovaný model

Vychází ze síťového modelu, který je doplněn o objekty z objektového programování. Základním stavebním kamenem modelu je objekt, který má svoje atributy a má definované své chování. [1, 2]

1.2.3.6 Cloud computing

„Cloud computing je model umožňující pohodlný síťový přístup na vyžádání ke sdílenému zásobníku konfigurovatelných výpočetních zdrojů (například sítí, serverů, paměti, aplikací a služeb), které mohou být rychle dodány a uvolněny s minimálním úsilím investovaným pro řídící činnosti či interakce s poskytovatelem služby.“ [3 s.239]

1.2.5.7 NOSQL

Využívá netradiční přístup k návrhu databáze, který ukládá data nerelačně. Namísto typické tabulkové struktury relační databáze, NOSQL ukládá data v jedné datové struktuře. Data mohou být v databázi rozložena nejen horizontálně, ale i vertikálně a jsou často organizována na logické úrovni do tabulek a přistupovaná jen za pomoci primárního klíče. Protože tato nerelační konstrukce databáze nevyžaduje schéma, nabízí rychlou škálovatelnost pro správu velkých a typicky nestrukturovaných datových souborů. [3]

1.2.4 Databázová aplikace

Je to počítačový program, který komunikuje s databází tak, že volá příkazy SQL pro DBMS. [6]

1.2.5 Architektury DBMS z hlediska zpracování

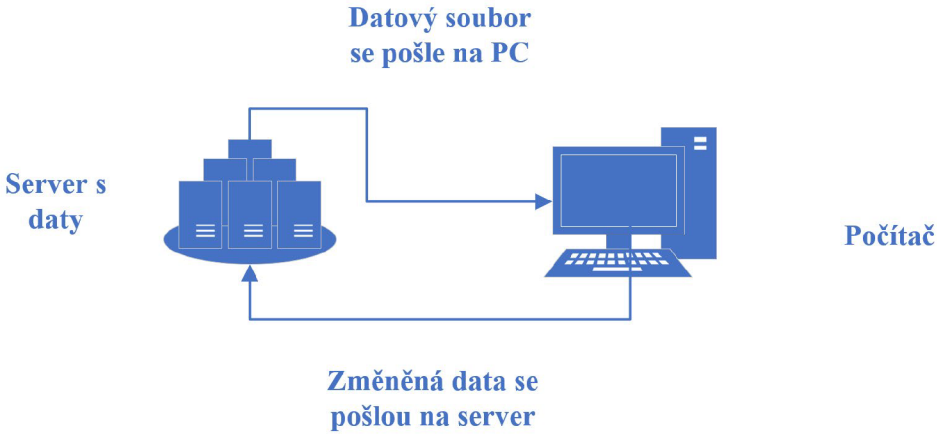
a) Centralizované platformy (jednovrstvá architektura)

Zastaralá architektura, kterou využívaly převážně sálové počítače (mainframe). Všechny součásti databáze (logika zpracování dat a provozu, databáze, datové služby) se prováděly na jednom počítači. Uživatelé mohli zobrazovat data na terminálech, které byly připojeny na sálový počítač. [2]

Obrázek č. 4: Centralizovaná platforma (Zdroj: [1] s.5)

b) Systémy na osobních počítačích

V 70. letech vznikaly první počítače, a to umožňovalo fungování DBMS a databázové aplikace na počítači místo na serveru. Počítač sloužil jako hostitelský počítač a zároveň jako terminál. Server sloužil jako datové úložiště a s počítačem byly na sebe připojeni pomocí síťového média (kabelem). Nevýhodou byla rychlost DBMS, která byla závislá na výkonu počítače a ne serveru. [2]



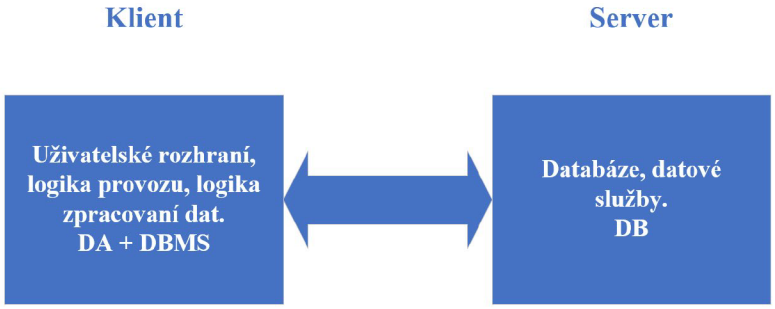
Obrázek č. 5: Systémy sálových počítačů (Zdroj: [2] s.7)

c) Databázový systém klient/server

Tato architektura rozděluje databáze zpracováním mezi dva systémy PC a server. Výhody této architektury jsou: rozdělení výpočetní zátěže, adaptabilita, transparentní přístup k datům, škálování systému a aplikací.

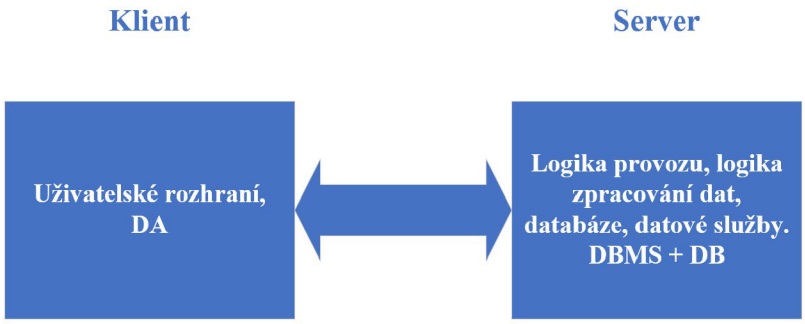
Typy této architektury:

- Architektura soustředěná u klienta – u klienta běží uživatelské rozhraní, logika provozu, logika zpracování dat.



Obrázek č. 6: Architektura soustředěná u klienta (Zdroj: [6])

- Architektura soustředěná na serveru – u klienta běží uživatelské rozhraní a logika provozu, logika zpracování dat, databáze a služby jsou soustředěny na serveru.



Obrázek č. 7: Architektura soustředěná na serveru (Zdroj: [6])

- V praxi se využívá třívrstvá architektura, která rozděluje služby na vrstvu uživatelského rozhraní, která běží u klienta, logika provozu a logika zpracování dat, která běží na aplikačním serveru a databázi a datové služby, které běží na databázovém serveru.

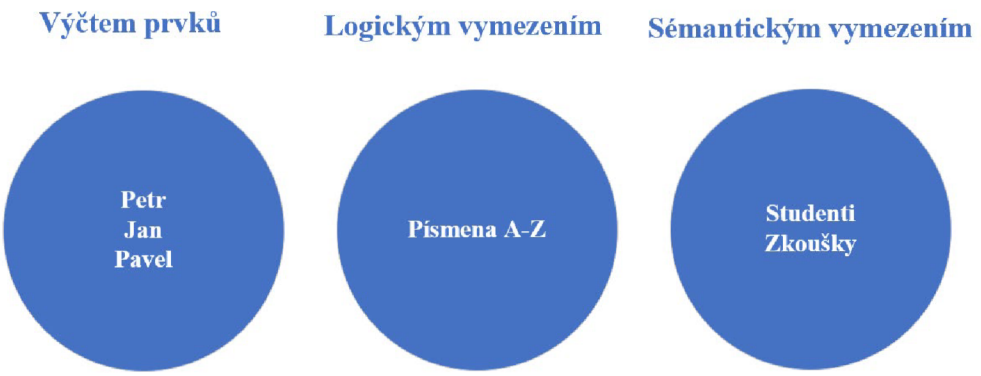
Obrázek č. 8: Třívrstvá architektura (Zdroj: [6])

Hlavním cílem tříúrovňové architektury je fyzická a logická nezávislost dat. Logická nezávislost znamená, že změna konceptuálního schématu (úprava entity, relace,...) nutně nevyvolá změnu externího schématu (úrovně pohledů), nebo povinnost úpravy databázové aplikace. Fyzická nezávislost umožňuje změnu interního schématu bez toho, aby se muselo změnit konceptuální schéma. [2, 6]

1.3 Relační model

1.3.1 Teorie množin

Teorie množin slouží jako nástroj pro definici a popis prvků datového modelu. Množiny definujeme pomocí domén třemi způsoby:

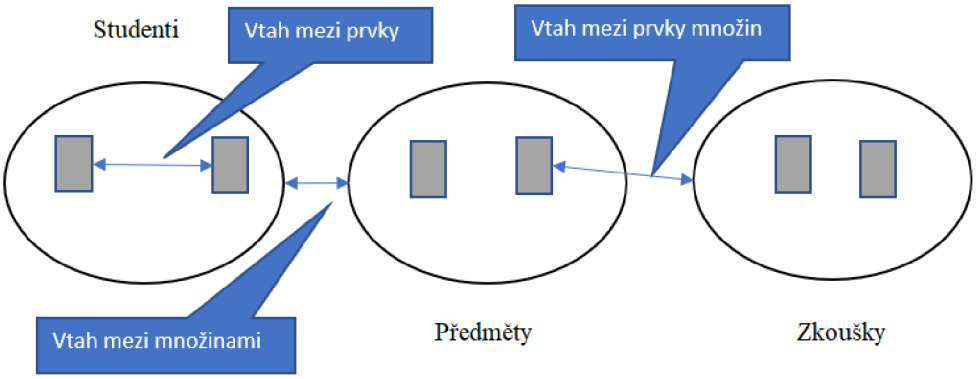


Obrázek č. 9: Způsoby definice prvků (Zdroj: [1] s. 24)

U množin rozlišujeme dva typy vztahů, a to vztahy mezi prvky množiny a vztahy mezi množinami (asociace). Asociace jsou mezi celými množinami nebo mezi prvky množin.

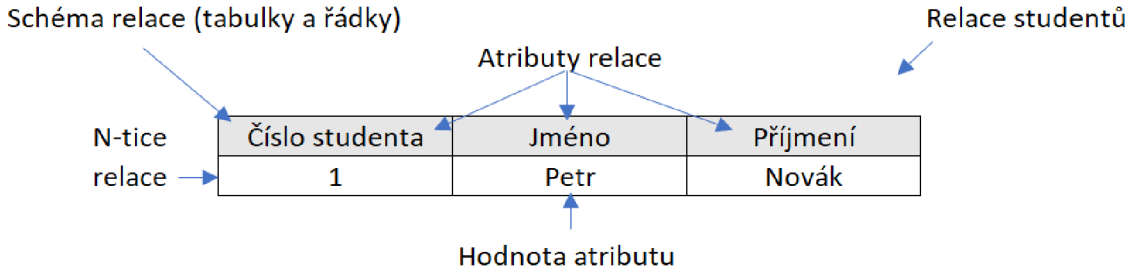
Pro práci s množinami se používají tyhle nástroje:

- Sjednocení množin.
- Průnik množin.
- Rovnost množin.
- Inkluze.
- Doplněk množiny. [1]



Obrázek č. 10: Vztahy (Zdroj: [1] s.25)

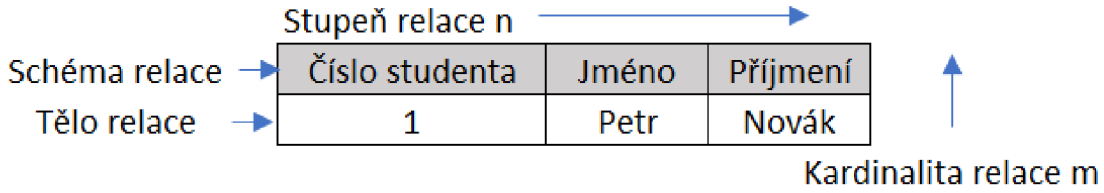
1.3.2 Terminologie z pohledu relací



Obrázek č. 11: Schéma relace (Zdroj: [1] s. 24)

„Máme-li množiny, v terminologii teorie množin „domény" například číslo studenta – D_1, jméno – D_2 a příjmení – D_3, pak relace na doménách D_1, D_2, …, D_n je dvojice $R = (R, R^*)$, kde $R = R(A_1:D_1, A_2:D_2, \ldots, A_n:D_n)$ je schéma relace a $R^* \subseteq D_1 \times D_2 \times \ldots \times D_n$ je tělo relace." [1, s. 25]

Tělo relace tvoří odpovídající podmnožina kartézského součinu. Schéma relace má zápis ve tvaru R (A_1 až A_n). Stupeň relace je počet atributů n-tic relace a kardinalita relace je kardinalita těla relace m = |R*|. Datová struktura relačního modelu se skládá z n-ární relace s pomocnou strukturou (schématem relace). [1]



Obrázek č. 12: Popis schématu relace (Zdroj: [1] s. 26)

1.3.3 Pravidla pro tabulkovou prezentaci relace

- Pořadí n-tic relací (řádků) a atributů (sloupců) je nevýznamné.
- Každá entita odpovídá jedné n-tici relace.
- Žádné n-tice relací nejsou stejné (žádné duplicity).
- Hodnoty atributů jsou atomické.
- Význam atributu je dán názvem atributu. [1]

1.3.4 Integrita

Jako každý model má relační model své teoretické omezení. Integritu můžeme definovat jako stav, při kterém data odpovídají vlastnostem objektů reálného světa. Rozlišujeme integritní omezení pro entity (relace) a pro vztahy entit (relační vazby). [1]

1.3.4.1 Integritní omezení pro entity

1) Doménová integrita
    - Určení povolených hodnot pro atribut.
    - Typ pole.
    - Musí obsahovat položky (NOT NULL).
    - Rozsah hodnot (maximální, minimální hodnota).
    - Implicitní hodnota.
    - Maska pro vkládání.
    - Seznam přípustných hodnot (číselník).
2) Entitní integrita

    Primární klíč (Primary key) – je to nezávislá množina atributů, která musí obsahovat vždy hodnotu a vyznačuje se dvěma vlastnostmi, kterými jsou:

1. Jednoznačnost – v relaci neexistuje atribut, který by měl stejné hodnoty jako má primární klíč.
2. Minimalismus – žádný atribut nelze vypustit bez toho, aby se porušila jednoznačnost.

Každá relace musí mít povinně určený primární klíč. Hodnoty primárního klíče jednoznačně identifikují instance (každou n-tici relace).

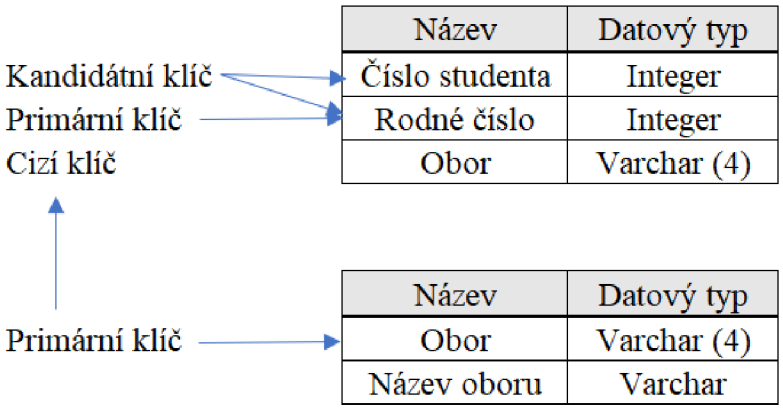
Kandidátní klíč (Candidate key) – má stejné vlastnosti jako primární klíč, ale nemusí být primární klíč. Umožňuje jednoznačně identifikovat instanci.

3) Referenční integrita

Cizí klíč (Foreign key) – je to atribut, který má tyto nezávislé vlastnosti:

1. Hodnota musí být plně zadaná nebo plně nezadaná.
2. Existuje primární klíč v relaci, který má identické hodnoty jako cizí klíč v jiné relaci.

Primární klíč s cizím klíčem tedy tvoří spojení mezi relacemi a platí zde pravidla referenční integrity (cizí a primární klíč musí mít nadefinované identické atributy a cizí klíč nemůže obsahovat jiné hodnoty než primární klíč). [1]



Obrázek č. 13: Referenční integrita (Zdroj: [1] s. 29)

1.3.4.2 Integritní omezení pro vztahy

a) Vztah 1:1

Jedna n-tice relace odpovídá jedné n-tici druhé relace. Například jeden člověk má 1 rodné číslo.

Obrázek č. 14: Vztah 1:1 (Zdroj: [1] s. 31)

b) Vztah 1:N

Jedna n-tice relace může odpovídat 1 nebo více n-tic druhé relace příkladem je relace mezi studentem a zkouškou, kde 1 student může vykonávat více zkoušek.

Obrázek č. 15: Vztah 1:N (Zdroj: [1] s. 31)

c) Vztah N:M

N n-tice relace odpovídá m n-tic druhé relace. Příkladem takového vztahu je student a předmět.

Obrázek č. 16: Vztah N:M (Zdroj: [1] s. 32)

Relace student

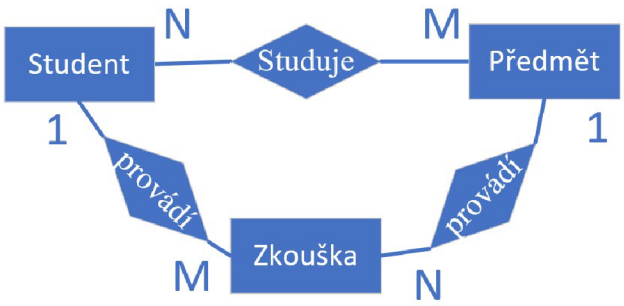
	Název	Datový typ
Primární klíč →	Číslo studenta	Integer
	Rodné číslo	Integer

Relace předmět

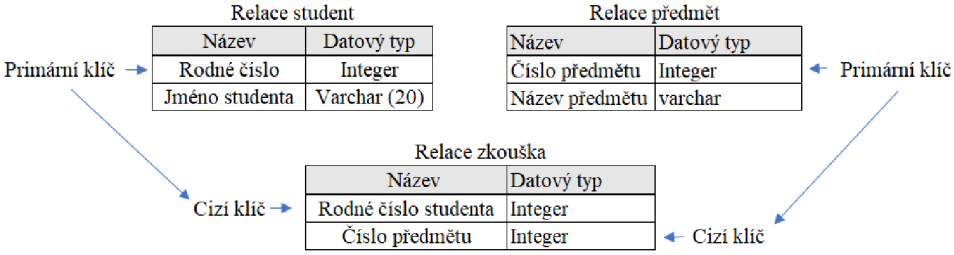
Název	Datový typ	
Číslo předmětu	Integer	← Primární klíč
Název předmětu	varchar	

Obrázek č. 17:Vztah N:M v relacích (Zdroj: [1] s. 34)

Pro tento vztah musíme provádět dekompozici, protože nemůžeme vést vazby mezi oběma entitami. To znamená, že musíme vytvořit novou relaci složenou z primárních klíčů relací.



Obrázek č. 18: Vztah N:M po dekompozici (Zdroj: [1] s. 34)



Obrázek č. 19: Vztah N:M po dekompozici v relacích (Zdroj: [1] s. 34)

1.3.5 Relační schéma databáze

Skládá se z dvojic (R, I), kde R je množina schémat relace, I je množina integritních omezení. Pokud relace splňují integritní omezení, data se stávají konzistentními. [1]

1.3.6 Jazyk SQL

Je to strukturovaný dotazovací jazyk, který vznikl v 70. letech společností IBM pro přístup k relačním databázím. SQL se používá pro definici a manipulaci s daty u všech databázových systémů. Velmi důležitou vlastností je podpora uložených procedur na databázovém serveru. Jsou to programy, které jsou zapsány přímo v samotné databázi (pokud je klient zavolá, provedou se celé na serveru), a tím zlepšují rychlost celé databáze. [1, 2]

1.3.6.1 Jazykové prostředky SQL

- Jazyk pro definici dat (DDL) se používá pro vytvoření, úpravu nebo smazání definic dat v databázi. Příkazy: CREATE, ALTER, DROP. [3]

- Jazyk pro manipulaci dat (DML) slouží k vložení, modifikaci a odstranění dat v databázi. Další činností je dotazování, které komunikuje přes schéma databáze. Příkazy: SELECT, INSERT, UPDATE, DELETE. [3]
- Jazyk pro transakce (TCC) zajišťuje integritu databáze tak, aby se operace v databázi provedly úplně nebo vůbec. Příkazy: BEGGIN, COMMIT, ROLLBACK. [7]
- Jazyk pro řízení dat (DCL) zabezpečuje databázi omezeným přístupem uživatelů za pomocí práv. Příkazy: GRANT, REVOKE. [8]

1.3.6.2 Datové typy SQL

a) Celá čísla

- TINYINT – hodnoty od -128 až po 127.
- SMALLINT – hodnoty od -32 768 až po 32767.
- MEDIUMINT – hodnoty od -8 388 608 až po 8 388 607.
- INT/INTEGER – hodnoty od -2 147 483 648 do 2 147 483 647 nebo od 0 po 4 294 967 295.
- BIGINT – hodnoty od -2^63 po 2^63-1.
- BIT – hodnoty od 0 po 1.

b) Čísla s pohyblivou desetinou čárkou

- FLOAT – hodnoty od -1.79E+308 do 1.79E+308.
- REAL – hodnoty od -3.4E+38 do 3.40E +38.
- DECIMAL (m, d) – rozsah hodnot si určujeme pomocí parametrů m, d.

c) Datum a čas

- DATE – datum ve formátu rok, měsíc a den (RRRR-MM-DD).
- DATETIME – datum a čas (RRRR-MM-DD HH-MM-SS).
- TIME – čas (HH-MM-SS).
- YEAR – rok (RRRR).

d) Text

- CHAR (m) – řetězec m má rozsah 0-255 znaků (pokud je řetězec kratší, než je nastavená délka, nevyplněné znaky se vyplní mezerami).

- VARCHAR (m) – řetězec m má proměnnou délku (zde se nevyplňují chybějící znaky), a navíc se ukládá informace o jeho délce.
- TYNYBLOB – řetězec má maximálně 255 znaků.
- BLOB – řetězec má maximálně 65 535 znaků.

e) **Logická**

- Bit – nabývá hodnot 0 nebo 1. [1, 2]

1.3.7 Entito-relační model (ER)

K vytvoření ER modelu je potřeba systematická analýza a cílem je definovat entity, vztahy, atributy a relace. ER diagramy slouží k následujícím účelům:

- Modelují informační požadavky organizace.
- Identifikují entity a jejich vztahy.
- Poskytují výchozí bod pro definici dat (diagramy datových toků).
- Poskytují vynikající zdroj dokumentace pro vývojáře aplikací a správce databází i systémů.
- Vytvářejí logický návrh databáze, který lze převést do fyzického schématu.

ER model se znázorňuje graficky nebo slovně. [9]

1.3.8 Normalizace

Úkolem normalizace je úprava uložených dat podle určitých pravidel pro efektivní ukládání dat a minimalizaci redundance při zachování integrity a konzistence dat. Pokud datový model porušuje jednu z forem normalizace, tak je neoptimálně navržený. Normalizace je dekompozice relací do optimálnější struktury tak, aby byla zachována bezztrátovost, závislost a odstraněny duplicity. Existuje 5 normálních forem.

1. **Normální forma (multizávislost)**

   Všechny atributy entit musí být jednoduché. To znamená že nesmí být vícehodnotové nebo složené, z důvodu přehlednosti a jednoduchosti. Příkladem je atribut adresa, která se po 1. normální formě rozloží na tyto jednotlivé dílčí (atomické) atributy: ulice, číslo popisné, město, obec, PSČ.

2. **Normální forma (funkčně závislá)**

Pro splnění druhé normální formy musí relace splňovat první normální formu a všechny její atributy jsou závislé na primárním klíči.

To znamená, že musíme prozkoumat všechny atributy a zjistit, zda závisí na primárním klíči. Například v tabulce (relaci) níže vidíme, že jméno studenta logicky závisí na jeho rodném číslu.

Relace student

	Název	Datový typ
Primární klíč →	Rodné číslo	Integer
	Jméno studenta	Vaarchar (20)

Obrázek č. 20: 2. normální forma (Zdroj: [1] s. 56)

3. **Normální forma (tranzitivní závislost)**

Musí splňovat první a druhou normální formu, a k tomu každý neklíčový atribut nesmí být tranzitivně závislý, to znamená, že atribut musí být funkčně závislý na primárním klíči a nezávislý na jiném neklíčovém atributu.

Zde máme tabulku (relaci) student, kde atribut fakulta je závislý na rodném číslu studenta, ale rodné číslo je závislé jen zprostředkovaně přes neklíčový atribut fakulty.

Relace student

Název	Datový typ
Rodné číslo	Integer
Jméno studenta	Varchar (20)
Fakulta	Integer

Obrázek č. 21: 3. normální forma před dekompozicí (Zdroj: [1] s. 58)

Proto uděláme dekompozici, a tím nám vznikne nová tabulka (relace) fakulta.

Relace student

	Název	Datový typ
Primární klíč →	Rodné číslo	Integer
	Jméno studenta	Varchar (20)
← Cizí klíč	Fakulta	Integer

Relace fakulta

	Název	Datový typ
Primární klíč →	Fakulta	Integer
	Název fakulty	Varchar

Obrázek č. 22: 3. normální forma po dekompozici (Zdroj: [1] s. 58)

Boyce – Coddova normální forma

Je to variace třetí normální formy a platí, že pokud je relace v Boyce-Coddově formě, tak splňuje i třetí normální formu, ale naopak to neplatí. Aby relace byla v Boyce-Coddově normální formě musí splňovat tyto podmínky:

- Mezi kandidátními klíči není žádná funkční závislost.
- Relace obsahuje dva nebo více kandidátních klíčů.
- Alespoň 2 z kandidátních klíčů musí být složené.
- V atributech se kandidátní klíče musí překrývat.

4. **Normální forma**

   Relace musí být v Boyce-Coddově normální formě a všechny vícehodnotové závislosti jsou i funkčními závislostmi kandidátních klíčů, aby byla ve čtvrté normální formě.

5. **Normální forma**

   S pátou normální formou se potkáme jen výjimečně a týká se situace spojené závislosti. Relace je v páté normální formě, pokud je ve čtvrté normální formě a nemůže být rozložena bez ztráty dat. [1, 2]

1.4 SQL

1.4.1 Vývojové prostředí Microsoft SQL server management studio (SSMS)

SQL Server Management Studio je integrované prostředí pro správu jakékoli infrastruktury SQL. SSMS poskytuje nástroje pro konfiguraci, monitorování a správu instancí SQL Serveru a databází. Pomocí SSMS můžeme nasazovat, monitorovat a aktualizovat komponenty datové vrstvy používané aplikacemi a vytvářet dotazy a skripty. Pomocí SSMS můžete zadávat dotazy, navrhovat a spravovat databáze.

1.4.2 Funkce SQL

SQL poskytuje vestavěné funkce řazené podle kategorie, které můžeme používat v databázích SQL. Můžeme použít vestavené funkce, nebo si vytvořit vlastní uživatelsky definované funkce.

1. **Agregační funkce**

Agregátní funkce provádějí výpočet hodnot a vracejí jednu hodnotu. Agregační funkci můžete použít v kombinaci s klauzulí GROUP BY pro výpočet agregace na kategorie řádků.

Všechny agregační funkce jsou deterministické, což znamená, že při spuštění na stejných vstupních hodnotách vracejí vždy stejnou hodnotu.

2. Analytické funkce

Analytické funkce počítají agregovanou hodnotu na základě skupiny řádků. Na rozdíl od agregačních funkcí však mohou analytické funkce vracet více řádků pro každou skupinu. Analytické funkce můžete použít k výpočtu klouzavých průměrů, průběžných součtů, procent nebo výsledků top-N v rámci skupiny.

3. Funkce řazení

Funkce řazení vracejí hodnotu řazení pro každý řádek v oddílu. V závislosti na použité funkci mohou některé řádky získat stejnou hodnotu jako jiné řádky. Funkce řazení jsou nedeterministické.

4. Řádkové funkce

Funkce Rowset vrací objekt, který lze použít jako odkaz na tabulku v příkazu SQL.

5. Skalární funkce

Pracují s jednou hodnotou a poté vracejí jednu hodnotu. Skalární funkce lze použít všude tam, kde platí příslušný výraz.

Tabulka č. 2: kategorie skalárních funkcí (Zdroj: Vlastní zpracování dle: [11])

Kategorie funkcí	**Popis**
Konfigurační funkce	Vrátí informace o aktuální konfiguraci.
Převodní funkce	Podporuje odlévání a převod datových typů.
Kurzorové funkce	Vrátí informace o kursorech.
Datové typy a funkce pro datum a čas	Provádí operace se vstupními hodnotami data a času a vrací řetězcové, číselné nebo datumové a časové hodnoty.
Funkce JSON	Ověřuje, dotazuje se nebo mění data JSON.
Logické funkce	Provádí logické operace.
Matematické funkce	Provádí výpočty na základě vstupních hodnot zadaných jako parametry funkcí a vrací číselné hodnoty.
Funkce metadat	Vrátí informace o databázi a databázových objektech.

Bezpečnostní funkce	Vrátí informace o uživatelích a rolích.
Řetězcové funkce	Provádí operace se vstupní hodnotou řetězce (char nebo varchar) a vrací řetězcovou nebo číselnou hodnotu.
Systémové funkce	Provádí operace a vrací informace o hodnotách, objektech a nastaveních v instanci serveru SQL Server.
Systémové statistické funkce	Vrátí statistické informace o systému.
Textové a obrazové funkce	Provádí operace nad vstupními hodnotami nebo sloupci textu nebo obrázku a vrátí informace o hodnotě.

6. Determinismus funkce

Vestavěné funkce SQL Serveru jsou buď deterministické, nebo nedeterministické. Funkce jsou deterministické, pokud při každém svém volání vracejí vždy stejný výsledek s použitím určité sady vstupních hodnot. Funkce jsou nedeterministické, když mohou při každém volání vrátit jiný výsledek, a to i při použití stejné konkrétní sady vstupních hodnot.

7. Slučování funkce

Funkce, které přijímají vstupní řetězec znaků a vracejí výstupní řetězec znaků, používají pro výstup collation vstupního řetězce. [11]

1.4.3 Základní informace pro dotazování v SQL

Pro základní dotazování nám stačí znát příkaz SELECT, který vybere seznam polí a FROM, pro vybrání, z jakých tabulek se budou brát data. V případě složitějších dotazů musíme znát agregaci a seskupení. Pomocí agregace SQL můžeme získat souhrnné údaje pomocí aritmetického průměru (AVG), součtu (SUM), maxima (MAX), minima (MIN) počtu řádků (COUNT) a pomocí seskupovaní se hodnoty seskupují podle sloupců, které vybereme. Sem patří příkazy GROUP BY (seskupuje sloupce) a ORDER BY (seřadí sloupce). K filtrování záznamů se používá klauzule WHERE, která vybere záznamy, které splňují zadanou podmínku. Další důležitou funkcí WHERE je spolu se spojením (JOIN) slučování tabulek (propojení dat ze dvou a více tabulek) a spojení JOIN se dále rozděluje na:

- Vnitřní spojení (INNER JOIN) – slučuje záznamy podle shody společných polí.

- Vnější spojení (OUTER JOIN) – dělí se na LEFT a RIGHT a podle strany, kterou vybereme přidá záznamy.
- Plné spojení (FULL JOIN) – vypíše záznamy se shodnými údaji, a k tomu přidá i záznamy z obou stran.
- Křížové spojení (CROSS JOIN) – spojuje úplně každý záznam z jedné strany s každým záznamem z druhé strany. [1, 2]

1.4.4 Syntaxe příkazů

Vytvoření tabulek

K vytvoření tabulek se používá příkaz CREATE TABLE. [2]

```
CREATE TABLE <Název tabulky> (<Název sloupce> <datový typ> <Integritní omezení>)
```

Manipulace s tabulkami

Pro změnu struktury tabulek se používá příkaz ALTER TABLE ADD, pro změnu hodnot se používá UPDATE, pro mazání řádků se používá DELETE, pro smazání tabulky se používá DROP TABLE a pro vložení hodnot INSERT INTO. [2]

```
ALTER TABLE <Název tabulky> ADD <Název sloupce> <Nový datový typ>
UPDATE <Název tabulky> SET <Název sloupce> = <hodnota> WHERE <Podmínka>
DELETE FROM <Název tabulky> WHERE <Podmínka>
DROP TABLE <Název tabulky>
INSERT INTO <Název tabulky> <(Názvy sloupců)> VALUES <(Hodnoty)>
```

Dotazy s agregací

Dotazy neboli SELECT slouží pro vybrání a zobrazení dat z tabulek podle námi nastavených kritérií. [2]

```
SELECT <Názvy sloupců> FROM <Název tabulek>
JOIN <Spojovací podmínka
WHERE <Výběrová kritéria>
GROUP BY <Sloupce k seskupení>
ORDER BY <Sloupce k seřazení>
```

Uložené procedury

Procedura je skript, který je uložený přímo v databázi a volá se prostřednictvím jejího názvu. [2]

```
CREATE PROCEDURE <Název>
<Název parametru, datový typ, výchozí hodnota, output>
AS
<programoví kód>
GO
```

Kurzory

Kurzor pracuje se záznamy postupně (pracuje s nimi jeden po druhém). Oproti příkazu SELECT se odlišuje těmito vlastnostmi: deklaruje se odděleně od jeho vlastního vykonání, kurzor má své pojmenování, výsledky v kurzoru zůstanou do té doby, než je nezavřeme a má speciální příkazy pro procházení záznamů.

V 1. kroku se deklaruje kurzor, v 2. kroku se provedou SQL příkazy a ve 3. kroku se data načtou a uloží do proměnných. [2]

```
1. DECLARE <název> CURSOR FOR SELECT <seznam polí> FROM <tabulka>
2. OPEN <Název>
3. FETCH <Název
```

Transakce

Transakce je soubor několika dotazů, které databáze chápe jako jeden. Důležitou funkcí je, že se transakce provede úplně, nebo vůbec a systém to dá vědět uživateli. Další důležitou vlastností je, že udržují konzistenci údajů. Klasický příklad transakce je změna zůstatku na účtu v bance. [2]

```
BEGIN TRAN <název> - zahájení transakce
COMMIT TRAN <název> - potvrzení transakce
ROLLBACK TRAN <název> - Odvolání transakce
SAVE TRAN <název> - vytvoření uloženého bodu transakce
```

Pohledy

Pohled je virtuální tabulka, která vypadá jako normální tabulka (TABLE), ale neobsahuje data. Pohled je definován na základě jedné nebo více tabulek. Můžeme provádět následující příkazy: SELECT, INSERT, UPDATE, DELETE, DROP. Ale nemůžeme provádět UNION,

ORDER BY. Pohledy se používají pro zabezpečení uživatelů nebo pro sumarizaci dat z více tabulek. [2]

```
CREATE VIEW <Název>
SELECT <Názvy sloupců> FROM <Tabulka>
```

Spouštěče

Jsou to defacto uložené procedury s tím rozdílem, že se provádí automaticky v případě nadefinované události (INSERT, UPDATE a DELETE). Využívá se například pro zálohu měnících se dat (například archiv změn známek). [2]

```
CREATE TRIGGER <název>
ON <tabulka>
BEFORE/AFTER [INSERT/UPDATE/DELETE]
AS
<programoví kód>
```

1.4.5 Řízení konkurenčního přístupu

Řízení konkurenčního přístupu je protokol, který spravuje více operací s databází tak, aby se navzájem nerušili. Tento protokol funguje v případě čtení tabulek více než jednou transakcí, ale problém nastává při aktualizaci dat, kde vzniká nekonzistence dat. Cílem tohoto protokolu je tedy rozvrhnout transakce tak, aby nedošlo ke vzájemnému rušení. Nejjednodušším řešením by bylo povolit běh vždy jediné transakce, ale cílem DBMS je maximalizace zpracování (aby mohli transakce probíhat současně).

Pro takový problém se převážně používají metody zamykání. Transakce při upravování dat v tabulce tabulku zamkne, a tím odepře přístup dalším transakcím k tabulce. Tento zámek skončí až se transakce provede nebo se neprovede vůbec. Rozlišujeme sdílený zámek, který umožňuje transakci číst položky, ale nikoliv je aktualizovat a exkluzivní zámek, který umožňuje číst i aktualizovat. Zámky fungují následujícím způsobem:

1. Pokud tabulka není uzamčena jinou transakcí, bude ji přidělen zámek.
2. Pokud je tabulka uzamčena, DBMS zjistí, jestli je požadavek slučitelný s existujícím zámkem. Pokud je požadavek odmítnut, transakce musí počkat, dokud se neodemkne zámek.

3. Zámek platí, dokud ho transakce explicitně neuvolní (dokud se transakce neprovede). [4]

1.4.6 Indexování databáze

„Index je datová struktura, která umožňuje DBMS rychleji lokalizovat konkrétní záznamy v souboru, a tak zrychlit odezvu na uživatelské dotazy“ [4 s. 511]

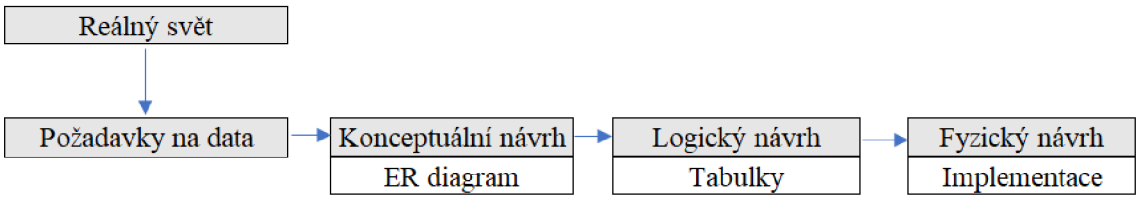
Index můžeme srovnat s rejstříkem v knize. Je to pomocná struktura spojená se souborem, která urychluje hledání položek. Index umožňuje vyhledávat položky, místo sekvenčně po záznamech (index seek), pomocí identifikátoru (idnex scan). Struktura indexu se skládá z vyhledávacího klíče, který obsahuje záznamy s hodnotou klíče a adresy logického záznamu v souboru, kde se hodnota nachází. Soubor, který obsahuje logické záznamy se nazývá datový soubor a soubor, který obsahuje záznamy o indexu se nazývá indexový soubor. V indexovém souboru jsou řazeny hodnoty podle indexačního pole a ten je založen na jednom sloupci. Typy indexů:

- Primární index – datový soubor je sekvenčně seřazený. Indexační pole je založeno na seřazeném klíčovém poli, který obsahuje jedinečné hodnoty každého záznamu.
- Seskupovací index – datový soubor je sekvenčně seřazený podle neklíčového pole. Indexační pole je také seřazeno podle neklíčového pole. Díky tomu může každé hodnotě indexu odpovídat více záznamů v datovém souboru. Neklíčové pole se nazývá seskupovací pole.
- Sekundární index – definuje se na základě nesetříděného pole datového souboru. [10]

```
Primární index: CREATE UNIQUE INDEX <Název indexu> ON <Názvy sloupců>
Seskupovací index: CREATE CLUSTER INDEX <Název indexu> ON <Názvy sloupců>
Zrušení indexu: DROP INDEX <Název indexu>
```

1.5 Metodologie návrhu databáze

Metodologie návrhu databáze je strukturovaný postup, za pomocí technik, nástrojů a dokumentace, pro usnadnění procesu návrhu.



Obrázek č. 23: Metodologie návrhu databáze (Zdroj: [6])

1.5.1 Životní cyklus vývoje DBS

Plánování databáze – tuto fázi má na starost management a hlavním úkolem je stanovení cílů a dílčích cílů, odhad potřebných zdrojů a finančních prostředků a integraci do strategie ICT v organizaci.

Definice systému – musí se definovat rozsah databázového systému a jeho integritu s částmi informačního systému organizace a co se od databázového systému požaduje z pohledu uživatele nebo oddělení organizace.

Sběr a analýza požadavků – pro vytvoření databáze musíme provést důkladnou analýzu prostředí pro získání potřebných informací, abychom určili požadavky na databázový systém.

Návrh databáze – vytvoření konceptuálního, logického a fyzického návrhu databáze. Souběžně se navrhuje databázová aplikace a vybírá se DBMS.

Vytvoření prototypu – vytvoření prototypu modelu databázového systému.

Implementace – fyzické vytvoření databáze a aplikace.

Konverze a načtení dat – úprava dat do vhodné podoby pro databázi a načtení dat.

Testování – testování databáze za účelem zjištění chyb.

Provoz a údržba – monitorování databáze a údržba celého databázového systému. [4, 6]

1.5.2 Konceptuální návrh databáze

Na základě uživatelských požadavků se vytvoří konceptuální návrh a výsledkem by měl být ER diagram. Konceptuální návrh se skládá z 9 kroků:

1. **Identifikace entit** – z požadavků uživatelů se definují entity.
2. **Identifikace relací** – z požadavků uživatelů se definují relace a doplní vztahy a integritní omezení mezi relacemi.
3. **Identifikace atributů** – určení jména, datového typu, délky, popisu atributů.
4. **Určení domén atributu** – určení množin hodnot, ze kterých čerpají další atributy.
5. **Určení primárních a kandidátních klíčů** – vyberou se entity, které budou kandidátní klíče a primární klíč.

6. **Specializace/generalizace** – nepovinný krok, ve kterém se modelují podtřídy a nadtřídy.
7. **Kontrola redundance modelu** – musí se zkontrolovat vztah relací 1:1, kde mohlo dojít k chybě a odstranění redundantních relací.
8. **Kontrola uživatelských transakcí** – každá transakce by měla být zdokumentovaná a musíme zkontrolovat, zda ER model podporuje transakce. To se může zkontrolovat pomocí popisu transakce (kontrola entit, atributů a relací, které transakce vyžaduje), nebo sledování cesty transakce (cesta transakce v ER diagramu, která vede k dekompozici ER diagramu na další ER diagramy popisující transakci).
9. **Diskuse na konceptuálním schématem s uživateli** – kontrola konceptu s uživatelem. [4, 6]

1.5.3 Logický návrh databáze

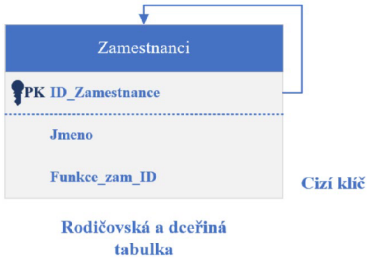
Logický návrh vychází z ER diagramu v konceptuálním návrhu. Hlavním cílem je vytvořit struktury tabulek a následně zkontrolovat jejich integritní omezení a normalizaci.

1. **Vytvoření tabulek** – zde se vytvoří tabulky pro entity a relace s atributy včetně integritních omezení. K tomuto kroku musíme znát typy relací, které jsem již představil, ale zde jsou doplněny o další typy a informace.

Binární relace jedna k více – relace 1:* (relace na straně 1 se označuje jako rodičovská a druhá relace na straně * jako dceřiná).

Obrázek č. 24: Binární relace 1:* (Zdroj: [6])

Rekurzivní relace jedna k více (1:*), (1:N) – vztahy jsou tvořeny uvnitř tabulky (unární relace). Prvky jsou hierarchicky poskládány, to znamená, že na první prvek jsou připojeny další prvky, které jsou o jednu úroveň níže.



Obrázek č. 25: Rekurzivní relace 1:* v relaci (Zdroj: [6])

Rekurzivní relace jedna k jedné – obě entity můžeme vložit do jediné tabulky, kde se jedna entita stane primárním klíčem a druhá alternativním.

Binární relace více k více – jak jsem již zmiňoval v části 1.3.4.2 musíme provést dekompozici, a tím vznikne nová tabulka složená z primárních klíčů ostatních relací.

Komplexní relace – jsou zde více jak dvě entity a řeší se to vytvořením nové tabulky, která obsahuje primární klíče entit. Tyto entity následně tvoří v nové tabulce složený primární klíč.

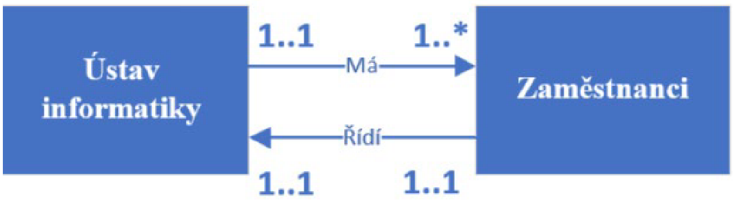
Binární relace jedna k jedné – kvůli tomu, že se nemůže jednoznačně určit pomocí kardinality, která tabulka je rodičovská nebo dceřiná, se musí použít participace. Ta rozhodne, jestli je vhodnější vytvořit pro obě entity vlastní tabulku nebo je spojit do jedné tabulky. Existují tři typy této relace:

a. **Povinná participace na obou stranách**

Obrázek č. 26: Povinná participace na obou stranách (Zdroj: [6])

Zde se můžou obě entity spojit do jedné tabulky. Jeden z primárních klíčů entit se stane primárním klíčem pro tabulku a druhý se stane alternativním.

b. Povinná participace na jedné straně



Obrázek č. 27: Povinná participace na jedné straně (Zdroj: [6])

Za pomocí integritního omezení participace rozhodneme, která tabulka bude rodičovská a dceřiná. Entita s povinnou participací je dceřiná a entita s nepovinou participací je rodičovská.

c. Nepovinná participace na obou stranách

Nelze rozhodnout na základě integritního omezení participace, proto je určení rodičovské a dceřiné tabulky na vlastním úsudku.

2. **Kontrola struktury tabulek** – kontrola vytvořených tabulek, zda splňují normalizaci alespoň třetí normální formy.
3. **Podpora uživatelských transakcí** – musí se zkontrolovat transakce z pohledu tabulek, zda nedochází k chybám.
4. **Kontrola integritních omezení** – ujištění správnosti integritních omezení pro konzistenci dat (omezení domén, integrita dat, referenční integrita, požadovaná data (NULL), multiplicita, …).
5. **Posouzení návrhu s uživateli** – kontrola konceptu s uživateli. [4, 6]

1.5.4 Fyzický návrh databáze

V této fázi by měl být schopný programátor vytvořit databázi na SQL serveru.

1. **Návrh a implementace tabulek** – konkrétní tvorba tabulek v databázi pomocí jazyka SQL.
2. **Návrh reprezentace odvozených položek** – nahrání dat do tabulek.
3. **Návrh zbývajících integritních omezení** – zajištění integritních omezení na serveru SQL například, aby rodné číslo bylo vždy zadáno.
4. **Organizace souborů a indexů** – kde se budou ukládat logovací soubory, datových souborů a indexových souborů.

5. **Návrh uživatelských pohledů** – vytvoření uživatelských pohledů.
6. **Návrh bezpečnostních mechanismů** – izolace od ostatních částí systému, určení práv pro programy, uživatele a další mechanismy.
7. **Kontrolované zavedení redundance** – porušení redundance při zavedení nové vazby za účelem zrychlení dotazu.
8. **Vyladění systému** – test databáze pro vyladění. [4, 6]

2 Analýza současného stavu

2.1 Představení společnosti

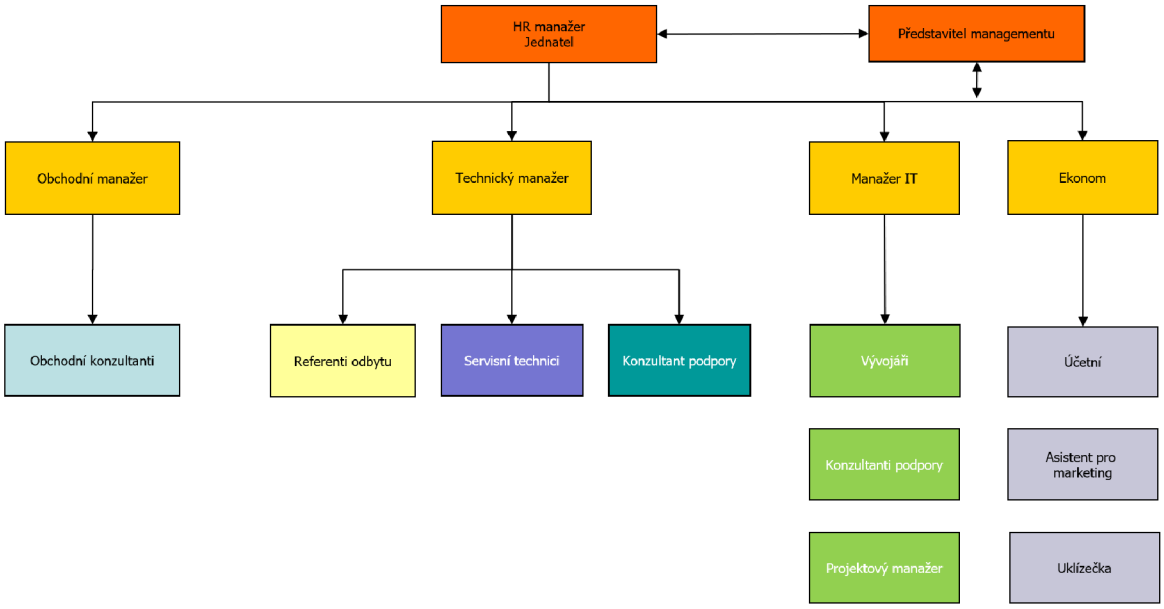
2.1.1 Základní údaje o společnosti

Bartech je česká společnost, která byla založena v roce 1995 a zaměřuje se na programování aplikací pro automatické identifikace, sběr dat, logistiku a řízení výroby. Společnost začínala vývojem softwaru pro menší firmy. Postupem času nasbírala dostatek zkušeností pro vytvoření vlastního softwaru jménem xTrace, a tím si i vytvořila jméno na trhu. Díky tomu získala tato firma zásadní spolupráci s velkými společnostmi, jako je IAC, Panasonic a Honeywell.

Obchodní firma:	**Bartech, s.r.o.**
Sídlo:	**Velkomoravská 527/33, 650 01 Hodonín**
IČO:	**26902931**
DIČ:	**CZ26902931**
Právní forma:	**společnost s ručením omezeným**
Den zápisu do obchodního rejstříku:	**7. srpen 2003**
Základní kapitál:	**200 000,- Kč**
Předmět podnikání:	**výroba, obchod a služby neuvedené v přílohách 1 až 3 živnostenského zákona**
Jednatel společnosti	**Ing. Antonín Měchura**
Logo:	BARTECH®

2.1.2 Organizační struktura

Společnost má liniovou organizační strukturu, která se skládá z jednatele společnosti, představitele managementu, 4 vedoucích manažerů a jim podřízených zaměstnanců. Všichni vedoucí i zaměstnanci pod vedoucími mezi sebou spolupracují a zodpovídají se jednateli nebo představiteli managementu.



Obrázek č. 28: Organizační struktura (Zdroj: interní dokumenty)

2.1.3 Software

Firma používá informační systém jménem myWAC. Ten běží na lokálním serveru a používá se na: sledování docházky, servis, podporu, prodej, úkoly a veškerou komunikaci se zákazníky. Dále je společnost partnerem Microsoftu, proto se programuje v Microsoft Visual studiu a pro databáze se používá Microsoft SQL Server Management Studio.

2.1.4 Hardware

Ve firmě se nachází 3 fyzické servery, které jsou rozděleny prostřednictvím softwaru VMWare na 17 virtuálních serverů. Na jednotlivé virtuální servery se může přiřadit potřebný počet hardwarových zdrojů. Serverové úložiště je řešeno pomocí diskového pole Synology, které využívá zálohu dat RAID 6. Dále má společnost fyzické switche, které jsou rozděleny na 5 virtuálních pomocí softwaru VMWare. 1. virtuální switch slouží pro intranet a nastavování sítě. 2. virtuální switch je pro NFS (pro vzdálený přístup k souborům na diskovém poli). 3. virtuální switch je pro připojení k internetu. 4. virtuální switch je pro DMZ (zabezpečení proti útokům zvenčí). 5. virtuální switch slouží pro Proxy. Specifikace serverů:

Server	Procesor	Soket	Jádra	Vlákna	Ram
HP ProLiant DL 360 G7	Intel® Xeon® CPU E5620 @2,40GHz	2	4	16	40
HP ProLiant DL360 G8	Intel® Xeon® CPU E5-2630 @2,30 GHz	2	6	24	98
Supermicro X8SIE	Intel® Xeon® CPU x3450 @2,67 GHz	1	4	8	25

Obrázek č. 29: Specifikace serverů (zdroj: vlastní zpracování)

2.1.5 Zabezpečení dat

Záloha dat je řešena pomocí diskového pole Synology DS620s, které má 6 disků, z toho je 5 disků určeno pro zálohu a jeden pro hotspare (záložní disk). Pro zálohování dat firma využívá software Active backup for bussines. Ve firmě jsou 2 typy záloh podle důležitosti dat – jednou denně a jednou týdně. Proti virům se využívá antivirus Windows Defender a firewall. Proti útokům zvenčí společnost využívá Proxy a DMZ.

2.2 Hlavní produkt firmy xTrace a výrobní proces

2.2.1 xTrace

xTrace je výrobní informační systém (MES) pro online sledování a řízení výroby včetně sběru dat ze strojů, zajištění plné zpětné dosledovatelnosti pro každý výrobek. Tento výrobní informační systém zajišťuje kontinuální sledování výroby v reálném čase, sběr dat ze strojů či jiných zařízení ve výrobě, komunikaci se stroji a ovládání dopravníků dle nastavených logik. Veškerá data jsou dlouhodobě uložena v databázi a následně využity pro výstup jako je kompletní rodný list výrobku nebo nástroje pro analýzu výroby za účelem zvýšení efektivity výroby.

2.2.2 Výrobní proces

Nastavení pracoviště

1. Přihlášení pracovníka

   xTrace vyžaduje vždy, aby byl na pracovišti přihlášen pracovník, který danou operaci provádí. Bez přihlášení není možné cokoliv na pracovišti dělat.

2. Načtení Jobu (výrobní objednávky)

   Samostatný proces v logice pracoviště, který následuje po přihlášení pracovníka. V tomto procesu se po pracovníkovi vyžaduje načíst čárový kód nebo ručně číslo výrobní objednávky, kterou bude na pracovišti vyrábět.

3. Výběr operace

   Když je vše v pořádku, pokračuje se vždy dalším krokem, a to je výběr operace z postupu, která se bude dělat.

4. Načtení nástrojů a komponent
   Načtou se nástroje a komponenty pro výrobu.

Výroba

5. Načtení sériového čísla dílu
   V tomto kroku se načítá sériové číslo hlavního dílu, na kterém se bude provádět zvolená operace.
6. Načtení sériových čísel připojovaných dílů

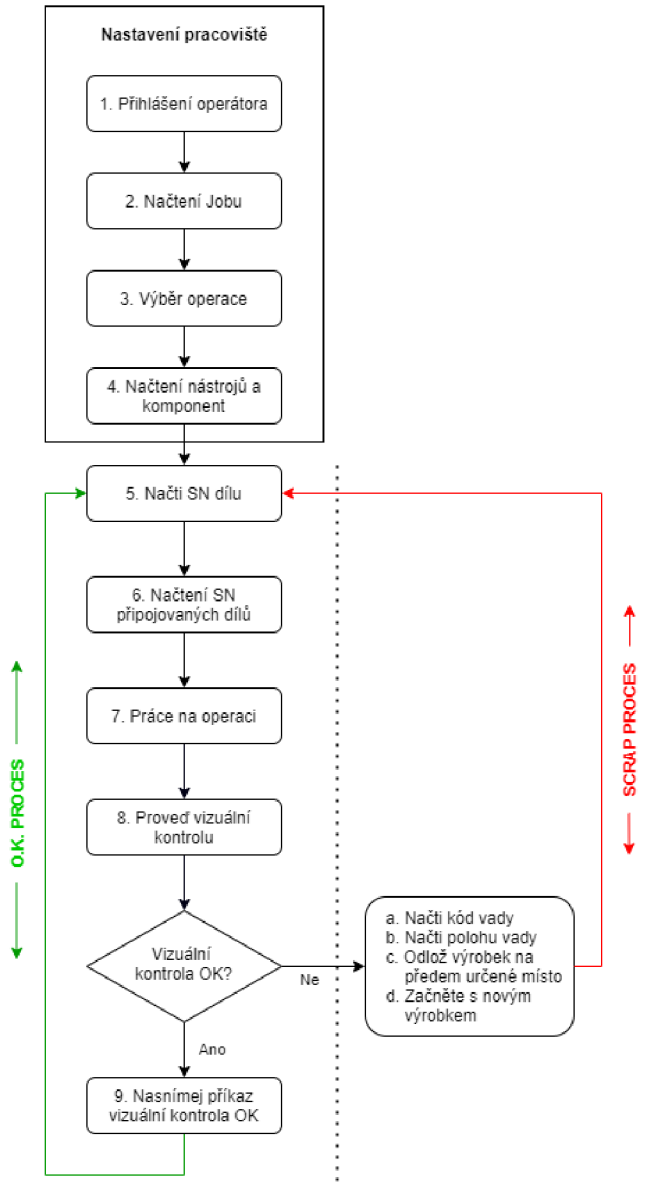
   Pokud je na dané operaci vydefinováno připojení jiného dílu, je operátor vyzván ho načíst.

7. Práce na operaci

   Samotný úkon, který je dán načtenou operací. Je to čas, který operátor potřebuje na provedení operace.

8. Provedení vizuální kontroly

V tomto kroku operátor zadává do systému, jak operace dopadla. Jsou dva stavy OK a NOK. Pokud je díl operátorem označený jako NOK, je dále standardním postupem určit vadu nasnímáním kódu z předlohy vad a následně určit polohu vady na daném díle. Tím dojde k ukončení dílu se stavem NOK a uložením vady a její pozice. Pokud je díl po operaci OK, potvrdí tento stav operátor načtením příkazového kódu. Tím se ukončí operace se stavem OK.



Obrázek č. 30: Diagram výroby (Zdroj: interní dokumenty)

2.3 Analýza databáze

Databázový systém je typu klient – server s architekturou soustředěnou na serveru. Server je přímo připojený do diskového pole (DAS) pro rychlejší posílání dat. Každý zákazník má vlastní databázi ve své firmě pro funkci xTrace a z ní se posílají data do databáze v Bartechu. Ve společnosti Bartech je 17 virtuálních serverů, které jsou jednotlivě rozděleny mezi zákazníky a na každém serveru je databáze (každý zákazník má svůj server s databází). Každý měsíc jsou do databáze posílány data od zákazníka pro práci a zlepšování xTrace. Například pokud by chtěl zákazník novou funkci, tak programátoři v Bartechu použijí data z databáze zákazníka v Bartechu.

Databáze se skládá ze 4 databází:

c) Databáze s výrobními instrukcemi – postupy a metody k výrobě.
d) Databáze s produkčními daty – zde se evidují výrobní objednávky.
e) Produkční databáze – uchovává informace o výrobku a jeho dosledovatelnosti.
f) Databáze s přihlašovacími daty z pracovišť – obsahuje informace o zaměstnancích, kteří pracují na pracovišti.

Pro zachování know-how firmy a bezpečnosti mi byla poskytnuta jen část databáze, proto provedu analýzu jen na části produkční databáze, která se potýká s problémy.

Při vytváření databáze se nepostupovalo podle zavedených postupů (určení faktů, vytvoření konceptuálního, logického a fyzického návrhu), protože si to společnost nemohla dovolit, ať už z pohledu času nebo peněz. Proto se v databázi nachází tabulky, které porušují jednu ze tří normálních forem.

Databáze obsahuje mnoho záznamů, což způsobuje pomalejší dotazování. Dále obsahuje databáze vícero transakcí, jejichž rychlost provedení závisí na rychlosti dotazování. Proto při výrobě občas vzniká problém s uváznutím.

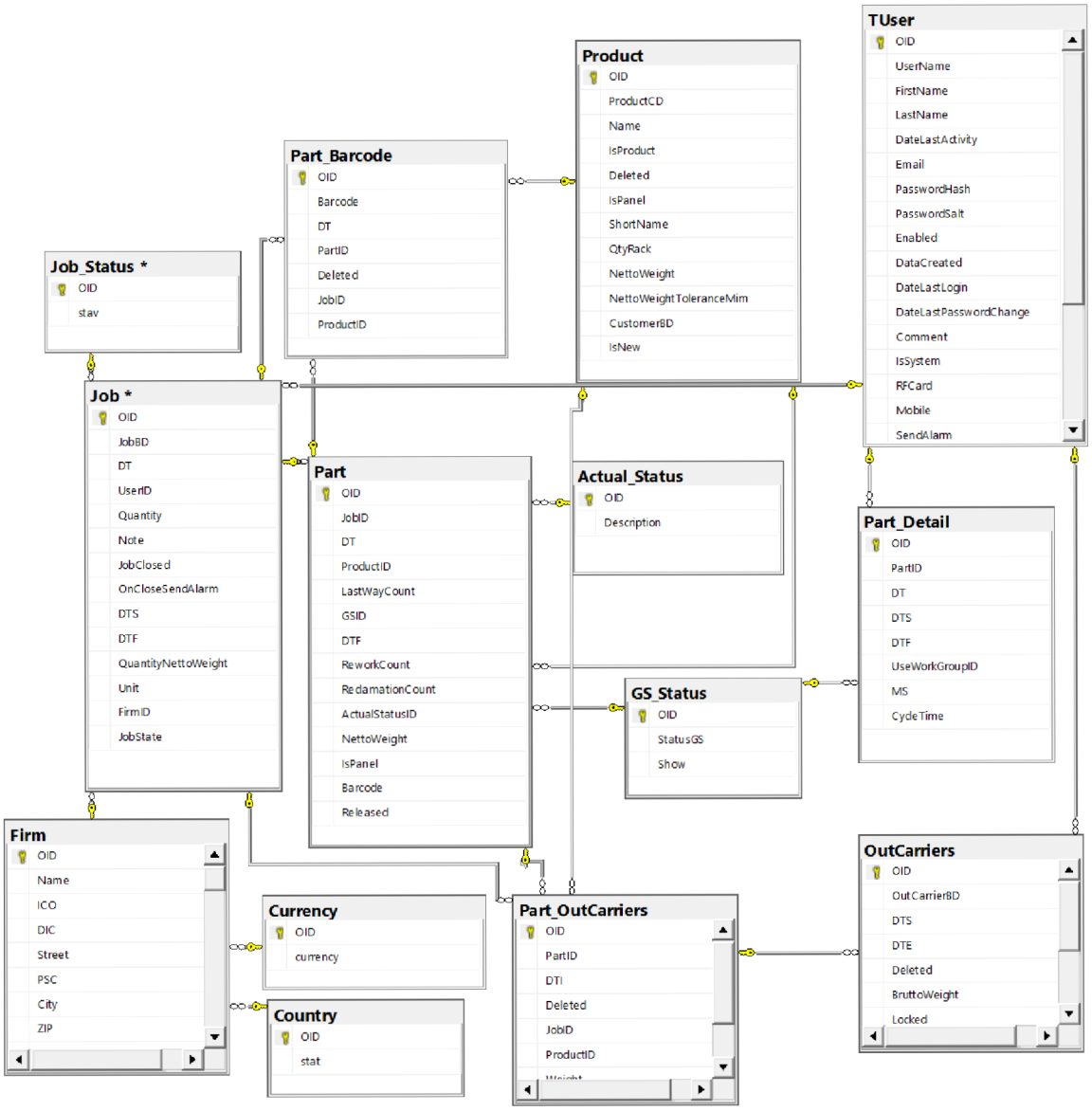
2.4 Zabezpečení databáze

Databázové servery se nachází v zabezpečené místnosti, do které má přístup jen technik ve firmě. Databáze je zabezpečena pomocí firewallu, který zajišťuje zabezpečení na úrovni operačního systému. Zabezpečení na úrovni databáze je řešená pomocí databázových rolí a ověření. Každý programátor má svoji roli a pro napojení do databáze se musí autorizovat svým uživatelským jménem a heslem.

Do databáze mají přístup jen programátoři ve firmě. Dále pro zabezpečení databáze mají omezená práva pro mazání tabulek, až na administrátora, který vlastní veškerá práva.

V Databázi není nastaveno jakékoliv šifrování dat. To znamená, že databáze není zabezpečená proti ukradnutí dat.

2.5 ER diagram databáze



Obrázek č. 31: ER diagram databáze (Zdroj: vlastní zpracování)

2.6 Analýza tabulek

V této části definuji tabulky v databázi. V tabulkách se nemažou neaktuální data pro uchování dohledatelnosti veškerých součástek. V případě, že data jsou neaktuální (například se již nevyrábí specifická součástka), obsahují tabulky sloupec Delete, který má datový typ bit (1, 0). Když se již nevyrábí tak sloupec Delete obsahuje hodnotu 1 a když se vyrábí tak 0.

2.6.1 Tabulka Part_Barcode

Evidují se zde sériová čísla dílů.

Part_Barcode			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID čárového kódu
	Barcode	Varchar(100)	Čárový kód/sériové číslo
	DT	Datetime	Kdy vznikl zápis do tabulky
Cizí klíč	PartID	Bigint	ID partu (součástky) z tabulky Part, atribut OID
	Deleted	bit	Logika, jestli je part_barcode smazany nebo ne
Cizí klíč	JobID	Bigint	ID jobu (operace) z tabulky Job, atribut OID
Cizí klíč	ProductID	Bigint	ID produktu z tabulky Product, atribut OID

Obrázek č. 32: Tabulka Part_Barcode (Zdroj: vlastní zpracování)

2.6.2 Tabulka Product

V této tabulce jsou veškeré informace o produktu.

Product			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID produktu
	ProductCD	Varchar(50)	Kód produktu
	Name	Varchar (240)	Název produktu
	IsProduct	Bit	Logika, jestli je produkt produkt nebo materiál
	Deleted	Bit	Logika, jestli je produkt smazany nebo ne
	IsPanel	Bit	Logika, jestli je produkt panelový
	ShortName	Varchar (50)	Zkratka produktu
	QtyRack	Int	Variabilní množství, kolik se vleze do nosiče
	NettoWeight	Decimal (18, 3)	Očekávaná váha finální produktu
	NettoWeight ToleranceMim	Decimal (18, 3)	Tolerance váhy produktu
	CustomerBD	Decimal (18, 3)	Zákaznický kód produktu
	IsNew	Bit	Logika, jestli je produkt nový

Obrázek č. 33: Tabulka Product (Zdroj: vlastní zpracování)

2.6.3 Tabulka Job

V této tabulce jsou veškeré informace o výrobních operacích.

Job			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID jobu (výrobní operace)
	JobBD	Bigint	Kód výrobního procesu
	DT	Datetime	Datum kdy byl vytvořen
Cizí klíč	UserID	Bigint	ID zaměstnance výroby z tabulky TUser, atribut OID
	Quantity	Int	Kolik se má vyrobit
	Note	Varchar (max)	Poznámka
	JobClosed	Bit	Logika, jestli je job hotový
	OnCloseSendAlarm	Bit	Logika, jestli se má zapnout alarm po dokončení výroby
	DTS	Datetime	Datum, kdy se začalo vyrábět
	DTF	Datetime	Datum, kdy se skončila výroba
	QuantityNettoWeight	Decimal (18, 3)	Váha celé výrobní objednávky
	Unit	Varchar (5)	V jakých jednotkách se vyrábí
Cizí klíč	FirmID	Bigint	ID firmy z tabulky Firm, atribut OID
Cizí klíč	JobState	Bigint	ID v jakém stavu se job nachází z tabulky Job_Status, atribut OID

Obrázek č. 34: Tabulka Job (Zdroj: vlastní zpracování)

2.6.4 Tabulka Part

Tabulka part uchovává veškeré informace o dílu.

Part			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID partu (součástky/dílu)
Cizí klíč	JobID	Bigint	ID jobu (výrobní operace) z tabulky Job, atribut OID
	DT	Datetime	Datum vzniku dílu
Cizí klíč	ProductID	Bigint	ID produktu z tabulky Product, atribut OID
	WayCount	Bigint	Kolikrát prošel pracovištěm
Cizí klíč	GSID	Bigint	ID stavu dílu ve výrobě z tabulky GS_Status, atribut OID
	DTF	Datetime	Datum dokončení
	ReworkCount	Int	Kolikrát prošla součástka opravou
	ReclamationCount	Int	Kolikrát byla součástka reklamována
Cizí klíč	ActualStatusID	Bigint	ID stavu součástky z tabulky Actual_Status, atribut OID
	NettoWeight	Decimal (18, 3)	Váha dokončené součástky
	IsPanel	Bit	Logika, jestli je součástka panelová
	Barcode	Varchar (100)	První čárový kód
	Released	Bit	Logika, jestli je vyřezaný panel z desky

Obrázek č. 35: Tabulka Part (Zdroj: vlastní zpracování)

2.6.5 Tabulka Part_OutCarriers

Tato tabulka obsahuje informace o dílech v jednom balení.

Part_OutCarriers			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID balení součástek
Cizí klíč	PartID	Bigint	ID součástky z tabulky Part, atribut OID
	DTI	Datetime	Datum vložení do krabice
	Deleted	Bit	Logika, jestli je balení smazané
Cizí klíč	JobID	Bigint	ID jobu z tabulky Job, atribut OID
Cizí klíč	ProductID	Bigint	ID produktu z tabulky Product, atribut OID
	Weight	Decimal (18, 3)	Váha
Cizí klíč	OutCarrierID	Bigint	ID balení z tabulky OutCarriers, atribut OID
	UserData	Varchar (512)	Poznámka

Obrázek č. 36: Tabulka Part_OutCarriers (Zdroj: vlastní zpracování)

2.6.6 Tabulka OutCarriers

Evidují se zde veškerá data o expedičním balení.

OutCarriers			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID balení
	OutCarrierBD	Varchar (30)	Čárový kód krabice
	DTS	Datetime	Datum vzniku balení
	DTE	Datetime	Datum expedice
	Deleted	Bit	Logika, jestli je produkt smazaný
	BruttoWeight	Decimal (18, 3)	Celková váha balení
	Locked	Bit	Logika pokud je balení zamčeno, s balením se nemůže nic provádět
Cizí klíč	UserWorkGroupID	Bigint	ID zaměstnance z tabulky TUser, atribut OID
	MaxQty	Int	Maximální počet balení

Obrázek č. 37: Tabulka OutCarriers (Zdroj: vlastní zpracování)

2.6.7 Tabulka TUser

Tabulka obsahuje informace o pracovnících výroby.

TUser			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID zaměstnance
	UserName	Varchar (50)	Přihlašovací jméno na web
	FirstName	Varchar (50)	Jméno
	LastName	Varchar (50)	Příjmení
	DateLastActivity	Datetime	Datum poslední aktivity
	Email	Varchar (50)	Email
	PasswordHash	Char (60)	Zákódované heslo
	PasswordSalt	Char (60)	Klíč k odkódování hesla
	Enabled	Bit	Logika, jestli je uživatel aktivní v systému
	DataCreated	Datetime	Kdy byl záznam vytvořen
	DateLastLogin	Datetime	Datum posledního přihlášení zaměstnance
	DateLastPasswordChange	Datetime	Datum poslední změny hesla
	Comment	Varchar (max)	Komentář
	IsSystem	Bit	Logika, jestli je to systemovy uživatel
	RFCard	Varchar (60)	Číslo karty
	Mobile	Varchar (30)	Telefoní číslo
	SendAlarm	Bit	Logika, jestli se maji posilat upozornění
	SendEmail	Bit	Logika, jestli se maji zasilat emaily
	CheckExpirationDate	Bit	Logika, jestli kontrolovat datum expirace
	ExpirationDate	Datetime	Datum splatnosti
	SendAlarmBeforeExpiration	Bit	Logika, pro zaslání upozornění před expirací
	EspirationAlarmSend	Bit	Logika, pro zaslání upozornění po expiraci
	CheckUserActivity	Bit	Logika, jestli je zaměstnanec aktivní

Obrázek č. 38: Tabulka TUSer (Zdroj: vlastní zpracování)

2.6.8 Tabulka Actual_Status

Tabulka obsahuje stav dílu.

Actual_Status			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID statusu
	Description	Varchar (25)	Stav dílu

Obrázek č. 39: Tabulka Actual_Status (Zdroj: vlastní zpracování)

2.6.9 Tabulka Part_Detail

Ukládají se zde výsledky operací na dílu.

Part_Detail			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID detailu součástky
Cizí klíč	PartID	Bigint	ID součástky z tabulky Part, atribut OID
	DT	Datetime	Datum zápisu do tabulky
	DTS	Datetime	Datum zahajení práce na pracovisti
	DTF	Datetime	Datum dokončení práce
Cizí klíč	UseWorkGroupID	Bigint	ID zaměstnance z tabulky TUser, OID
Cizí klíč	MS	Bigint	Stav součástky ve výrobě z tabulky GS_Status, atribut OID
	CycleTime	Datetime	Jak dlouho to trvalo. Rozdíl DTS a DTF

Obrázek č. 40: Tabulka Part_Detail (Zdroj: vlastní zpracování)

2.6.10 Tabulka Firm

Tabulka obsahuje základní informace o zákazníkovy.

Firm			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID firmy
	Name	Varchar (160)	Název firmy
	ICO	Varchar (15)	IČO
	DIC	Varchar (20)	DIČ
	Street	Varchar (80)	Ulice
	PSC	Varchar (10)	PSČ
	City	Varchar (40)	Město
	ZIP	Varchar (20)	ZIP
Cizí klíč	CountryID	Bigint	ID státu z tabulky Country, atribut OID
Cizí klíč	CurrencyID	Bigint	ID měny z tabulky Currency, atribut OID
	Email	Varchar (100)	Email
	WWW	Varchar (100)	Webové stránky
	Phone	Varchar (30)	Telefoní číslo
	Fax	Varchar (30)	Fax
	Contact	Varchar (30)	Jméno a příjmení kontáktní osoby
	ContactTitle	Varchar (20)	Titul kontaktní osoby
	Customer	Bit	Logika, jestli je zákazník
	Dealer	Bit	Logika, jestli je prodejce
	Suplier	Bit	Logika, jestli je dodavatel
	Transporter	Bit	Logika, jestli je dopravce
	Phone2	Varchar (30)	Druhé telefoní číslo
	Phone3	Varchar (30)	Třetí telefoní číslo
	Fax2	Varchar (30)	Druhý fax
	Fax3	Varchar (30)	Třetí fax
	Name2	Varchar (160)	Druhý název společnosti
	Street2	Varchar (80)	Druhá ulice
	City2	Varchar (40)	Druhé město
	PSC2	Varchar (10)	Druhé PSČ
	Email2	Varchar (100)	Druhý Email
	Email3	Varchar (100)	třetí Email
	WWW2	Varchar (100)	Druhá webová stránka
	WWW3	Varchar (100)	Třetí webová stránka
	Note	Varchar (1024)	Poznámka
	Deleted	Bit	Logika, jestli je firma smazana
	SubFirm	Bit	logika, jestli je firma dceřiná společnost
	Owner	Bit	Jestli je kontaktní osoba vlastníkem firmy
	ImageLogo	Image	Logo
	ImgType	Varchar (50)	Typ obrázku
	BankName	Varchar (50)	název banky
	BankAdress	Varchar (50)	Adresa banky
	BankAccount	Varchar (30)	Číslo účtu
	BIC	Varchar (30)	Mezinárodní kód banky
	IBAN	Varchar (30)	Mezinárodní kód banky
	WAT	Float	DIČ
Cizí klíč	Country2ID	Bigint	ID státu z tabulky Country, atribut OID
	FirmCD	Varchar (50)	Kod firmy

Obrázek č. 41: Tabulka Firm (Zdroj: vlastní zpracování)

2.6.11 Tabulka Currency

Tabulka obsahuje názvy měn.

Currency			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID měny
	currency	Varchar (25)	Název měny

Obrázek č. 42: Tabulka Currency (Zdroj: vlastní zpracování)

2.6.12 Tabulka GS_Status

Tabulka obsahuje názvy aktuálního stavu dílu ve výrobě.

GS_Status			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID stavu
	StatusGS	Varchar (25)	Stav dílu při výrobě
	Show	Int	Počet kusů při stavu

Obrázek č. 43: Tabulka GS_Status (Zdroj: vlastní zpracování)

2.6.13 Tabulka Country

Tabulka obsahuje názvy států.

Country			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID státu
	Stát	Varchar (250)	Název státu

Obrázek č. 44: Tabulka Country (Zdroj: vlastní zpracování)

2.6.14 Tabulka Job_Status

Tabulka obsahuje názvy stavu výrobního procesu.

Job_Status			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID statusu
	JobStatus	Varchar (150)	Stav výrobní operace

Obrázek č. 45: Tabulka Job_Status (Zdroj: vlastní zpracování)

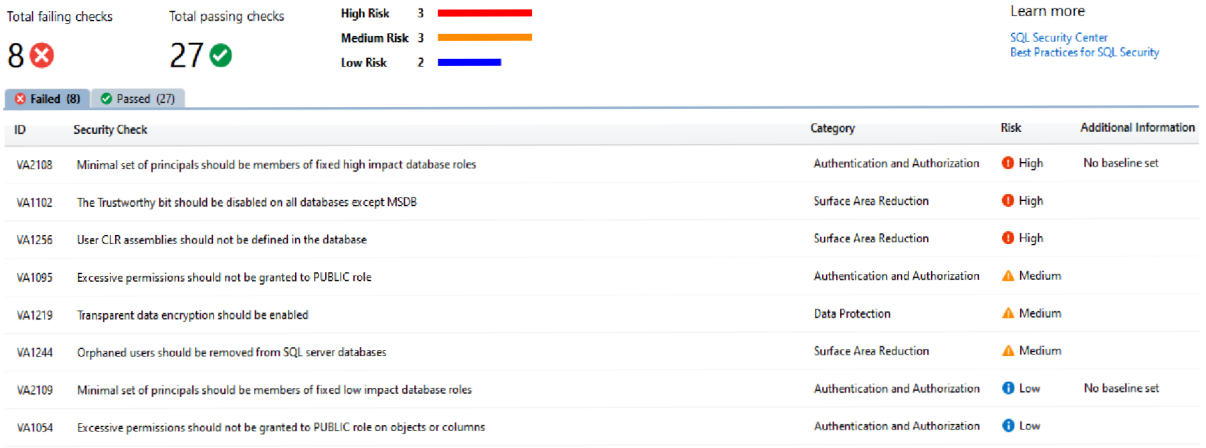
2.7 Analýza vztahů

Tabulka	Vztahy relací
Job - Part_Barcode	Binární 1:*, rodičovská tabulka je Job a dceřinná Part_Barcode
Job - TUser	Binární 1:*, rodičovská tabulka je TUser a dceřinná Job
Job - Firm	Binární 1:*, rodičovská tabulka je Firm a dceřinná Job
Currency - Firm	Binární 1:*, rodičovská tabulka je Currency a dceřinná Job
Country - Firm	Binární 1:*, rodičovská tabulka je Country a dceřinná Firm
Job - Part	Binární 1:*, rodičovská tabulka je Part a dceřinná Job
Job - Part_OutCarriers	Binární 1:*, rodičovská tabulka je Job a dceřinná Part_OutCarriers
Part - Part_OutCarriers	Binární 1:*, rodičovská tabulka je Job a dceřinná Part_OutCarriers
Part_Barcode - Part	Binární 1:*, rodičovská tabulka je Part a dceřinná Part_Barcode
Product - Part_Barcode	Binární 1:*, rodičovská tabulka je Product a dceřinná Part_Barcode
Part - Actual_Status	Binární 1:*, rodičovská tabulka je Acutal_Status a dceřinná Part
GS_Status - Part	Binární 1:*, rodičovská tabulka je GS_Status a dceřinná Part
Part - Product	Binární 1:*, rodičovská tabulka je Product a dceřinná Part
Part_OutCarrier - Product	Binární 1:*, rodičovská tabulka je Product a dceřinná Part_OutCarriers
Part_OutCarrier - OutCarriers	Binární 1:*, rodičovská tabulka je Part_OutCarriers a dceřinnáOutCarriers
GS_Status - Part_Detail	Binární 1:*, rodičovská tabulka je GS_Status a dceřinná Part_Detail
OutCarriers - TUser	Binární 1:*, rodičovská tabulka je TUser a dceřinná OutCarriers
Part_Detail - TUser	Binární 1:*, rodičovská tabulka je TUser a dceřinná Part_Detail
Job_Status - Job	Binární 1:*, rodičovská tabulka je Job_Status a dceřinná Job

Obrázek č. 46: Vztahy relací (Zdroj: vlastní zpracování)

2.8 Analýza bezpečnosti pomocí funkce „Posouzení zranitelnosti“ v SQL

Posouzení zranitelnosti SQL je služba, která poskytuje přehled o stavu zabezpečení a zahrnuje realizovatelné kroky k vyřešení bezpečnostních problémů a zvýšení zabezpečení databáze. [12]



Obrázek č. 47: Posouzení zranitelnosti SQL (Zdroj: vlastní zpracování)

2.9 Shrnutí analýzy

V databázi je pomalé dotazování kvůli absenci indexů. Díky tomu občas vznikají i uváznutí. Dále jsou v databázi tabulky, které porušují jednu ze tří normálních forem.

Využitím funkce posouzení zranitelnosti jsme zjistili nedostatky v bezpečnosti databáze, které je potřeba zanalyzovat a určit, jestli se jedná o opravdové riziko nebo ne.

Ve výsledku je celkový stav databáze až na zmíněné nedostatky v dobrém stavu. Databáze je velmi dobře zabezpečená na útoky zvenčí a má kvalitní zálohování.

3 Návrh zlepšení

Zde se budu zabývat návrhem řešení nedostatků databáze podle provedené analýzy.

3.1 Řešení rychlosti dotazování a uváznutí

Naindexování sloupců s cizími klíči, které jsou velmi využívané, zrychlí napojování záznamů a naindexování nejpoužívanějších sloupců zrychlí dotazování. Díky zrychlení dotazování a napojování se vyřeší i problém s uváznutím, protože se transakce budou provádět rychleji. Při indexování se musí brát v úvahu to, že indexy zabírají místo v databázi a mnoho indexů způsobí zpomalení výkonu databáze.

```
/*Part_Barcode*/
create unique nonclustered index Part_Barcode_Barcode on part_barcode (barcode)
create unique nonclustered index Part_Barcode_partID on part_barcode (partid)
/*Part_Detail*/
create unique nonclustered index Part_Detail_partid on part_detail (partid)
create nonclustered index Part_Detail_DT on part_detail (DT)
/*Part*/
create nonclustered index Part_JobID on Part (JobID)
create nonclustered index Part_DTF on Part (DTF)
create nonclustered index Part_ActualStatus on Part (ActualStatusID)
create unique nonclustered index Part_Barcode on part (barcode)
create nonclustered index Part_GSID on part (GSID)
create nonclustered index part_ProductID on Part (productid)
/*Job*/
create unique nonclustered index Job_JobBD on Job (JobBD)
create nonclustered index Job_UserID on Job (UserID)
/*Product*/
create nonclustered index Product_IsPanel on Product (IsPanel)
create nonclustered index Product_IsProduct on Product (IsProduct)
create unique nonclustered index Product_CustomerBD on Product (CustomerBD)
/*Part_OutCarriers*/
create nonclustered index Part_OutCarriers_JobID on Part_OutCarriers (JobID)
create unique nonclustered index Part_Outcarriers_OutCarrierID on
Part_OutCarriers (OutCarrierID)
create nonclustered index Part_Outcarriers_ProductID on Part_OutCarriers
(ProductID)
/*OutCarriers*/
Create unique nonclustered index OutCarrier_DTE_DTS on OutCarriers
(OutCarrierBD,DTE, DTS)
Create nonclustered index OutCarrier_UserWorkGroupID on OutCarriers
(UserWorkGroupID)
/*TUser*/
Create nonclustered index TUser_UserName on TUser (FirstName, LastName)
/*Firm*/
Create nonclustered index Firm_NameCompany on Firm (Name)
/*Actual_Status*/
Create unique nonclustered index Actual_Status_Description on Actual_Status
(Description)
```

```
/*GS_Status*/
Create unique nonclustered index GS_Status_StatusGS on GS_Status (StatusGS)
/*Job_Status*/
Create unique nonclustered index Job_Status on Job_Status (stav)
```

3.2 Normalizace a optimalizace tabulek

V této tabulce jsou rozepsané tabulky z databáze a jestli splňují 3 základní normální formy.

Tabulka č. 3: Tabulky dle splnění tří základních norem (Zdroj: Vlastní zpracování)

Tabulka	Normální formy		
	1. forma	2. forma	3. forma
Part_Barcode	Ano	Ano	Ano
Product	Ano	Ano	Ano
Job	Ano	Ano	Ano
Part_Barcode	Ano	Ano	Ano
Part_OutCarriers	Ano	Ano	Ano
OutCarriers	Ano	Ano	Ano
TUser	Ano	Ano	Ano
Firm	Ne	Ano	Ano
Part_Detail	Ano	Ano	Ano
Actual_Status	Ano	Ano	Ano
Currency	Ano	Ano	Ano
GS_Status	Ano	Ano	Ano
Country	Ano	Ano	Ano
Job_Status	Ano	Ano	Ano

3.2.1 Dekompozice tabulky Firm

Sloupec contact obsahuje složené jméno a příjmení, což porušuje 1. normální formu. Proto musíme tenhle sloupec rozložit na sloupce jméno a příjmení. První musíme přidat do tabulky Firm sloupce jméno a příjmení.

```
alter table firm
add Firstname varchar(15), Lastname varchar(20)
```

Pro rozložení sloupce contact a naplnění sloupců jména a příjmení využijeme tenhle kód. Logika skriptu je vytvořena na smyčce, která pro každé ID firmy rozloží jméno a příjmení ze sloupce contact pomocí funkcí substring a charindex, a následně naplní sloupce jméno a příjmení v tabulce.

```
declare @id int, @count int, @jmeno varchar(15), @prijmeni varchar(20)
set @count = 1
set @id = (select count(oid) from firm)
while @count<=@id
begin
set @jmeno = (SELECT SUBSTRING(contact, 1, CHARINDEX(' ', contact) - 1) from firm
where oid=@count)
set @prijmeni = (select SUBSTRING(contact, CHARINDEX(' ', contact) + 1,
LEN(contact) - CHARINDEX(' ', contact)) from firm where oid=@count)
update firm
set Firstname = @jmeno, Lastname = @prijmeni where oid=@count
set @count= @count +1
end
```

V případě, že chceme ukládat v databázi více než jeden telefon, musí se vytvořit nová tabulka pro telefon s relací jedna ku jedné, kde primární klíč ID firmy z tabulky Firm bude tvořit primární klíč v tabulce telefon, a zároveň bude cizím klíčem odkazující na ID firmy. Díky tomu můžeme přehledně vložit n telefonních čísel firmě.

Phone			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	int	ID uživatele
	Phone	Varchar (12)	Telefon
	Phone2	Varchar (12)	Druhý telefon

Obrázek č. 48: Tabulka phone (Zdroj: vlastní zpracování)

```
create table Phone (
OID bigint identity (1,1) primary key,
Phone varchar (12),
Phone2 varchar (12),
Foreign key (OID) references firm(OID))
```

Pro zkopírování telefonních čísel z tabulky Firm do tabulky Phone znova využijeme skript, který využívá smyčku, která pro každé ID telefonu vloží telefonní čísla firmy z tabulky Firm do tabulky Phone.

```
declare @id int, @count int, @telefon varchar(12), @telefon2 varchar(12)
set @count = 1
set @id = (select count(oid) from firm)
while @count<=@id
begin
set @telefon = (select phone from firm where oid=@count)
Set @telefon2 = (select phone2 from firm where oid=@count)
insert into phone (phone, phone2)
select phone, phone2 from firm where oid=@count
set @count= @count +1
end
```

3.2.2 Optimalizace tabulky Firm

Po konverzaci s vedením společnosti jsem zjistil, že se nevyužívají tyto sloupce: fax, fax2, fax3, name2, street2, city2, psc2, email2, email3, www2, www3, imgtype, bankname, bankadress, bankaccount, BIC, IBAN, WAT, currency, country2ID, proto je smažeme. K tomu smažeme i sloupce phone, phone2 (protože jsme je vložili do nové tabulky phone), contact (protože jsme ho rozložili na jméno a příjmení) a v poslední řadě tabulku currency, která nemá již využití.

Abychom mohli smazat sloupce currencyID a Country2ID musíme nejdříve smazat omezení. K tomu musíme zjistit název omezení tak, že klikneme v Microsoft SQL server managment studiu pravým tlačítkem myši na tabulku Firm, dále klikneme na design. To nám otevře schéma tabulky. Dále klineme znova pravé tlačítko na myši kdekoliv v tabulce a klikneme na Relationships. To nám otevře tabulku s omezením, kde najdeme názvy cizích klíčů currencyID a country2ID. Po zjištění názvů omezení v systému je smažeme.

```
ALTER TABLE firm
DROP CONSTRAINT FK__Firm__CurrencyID__4D94879B;
ALTER TABLE firm
DROP CONSTRAINT FK__Firm__Country2ID__4E88ABD4
```

Nyní nám už nic nebrání ve smazaní nepoužívaných sloupců a tabulky Currency.

```
Alter table firm
drop column phone, phone2, phone3, fax, fax2, fax3, name2, street2, city2, psc2,
email2, email3, www2, www3, imgtype, bankname, bankadress, bankaccount, bic,
iban, wat, currencyid, country2id, contact;
GO
Drop table currency
GO
```

3.2.3 Tabulka Firm po optimalizaci

Firm			
Integrita	**Atribut**	**Datový typ**	**Popis**
Primární klíč	OID	Bigint	ID firmy
	Name	Varchar (160)	Název firmy
	ICO	Varchar (15)	IČO
	DIC	Varchar (20)	DIČ
	Street	Varchar (80)	Ulice
	PSC	Varchar (10)	PSČ
	City	Varchar (40)	Město
	ZIP	Varchar (20)	ZIP
	Email	Varchar (100)	Email
	WWW	Varchar (100)	Webové stránky
	Firstname	Varchar (15)	Jméno kontáktní osoby
	Lastname	Varchar (20)	Příjmení kontáktní osoby
	ContactTitle	Varchar (20)	Titul kontaktní osoby
	Customer	Bit	Logika, jestli je zákaznik
	Dealer	Bit	Logika, jestli je prodejce
	Suplier	Bit	Logika, jestli je dodavatel
	Transporter	Bit	Logika, jestli je dopravce
	Note	Varchar (1024)	Poznámka
	Deleted	Bit	Logika, jestli je firma smazana nebo ne
	SubFirm	Bit	logika, jestli je firma dceřiná společnost
	Owner	Bit	Jestli je kontaktní osoba vlastníkem firmy
	FirmCD	Varchar (50)	Kod firmy

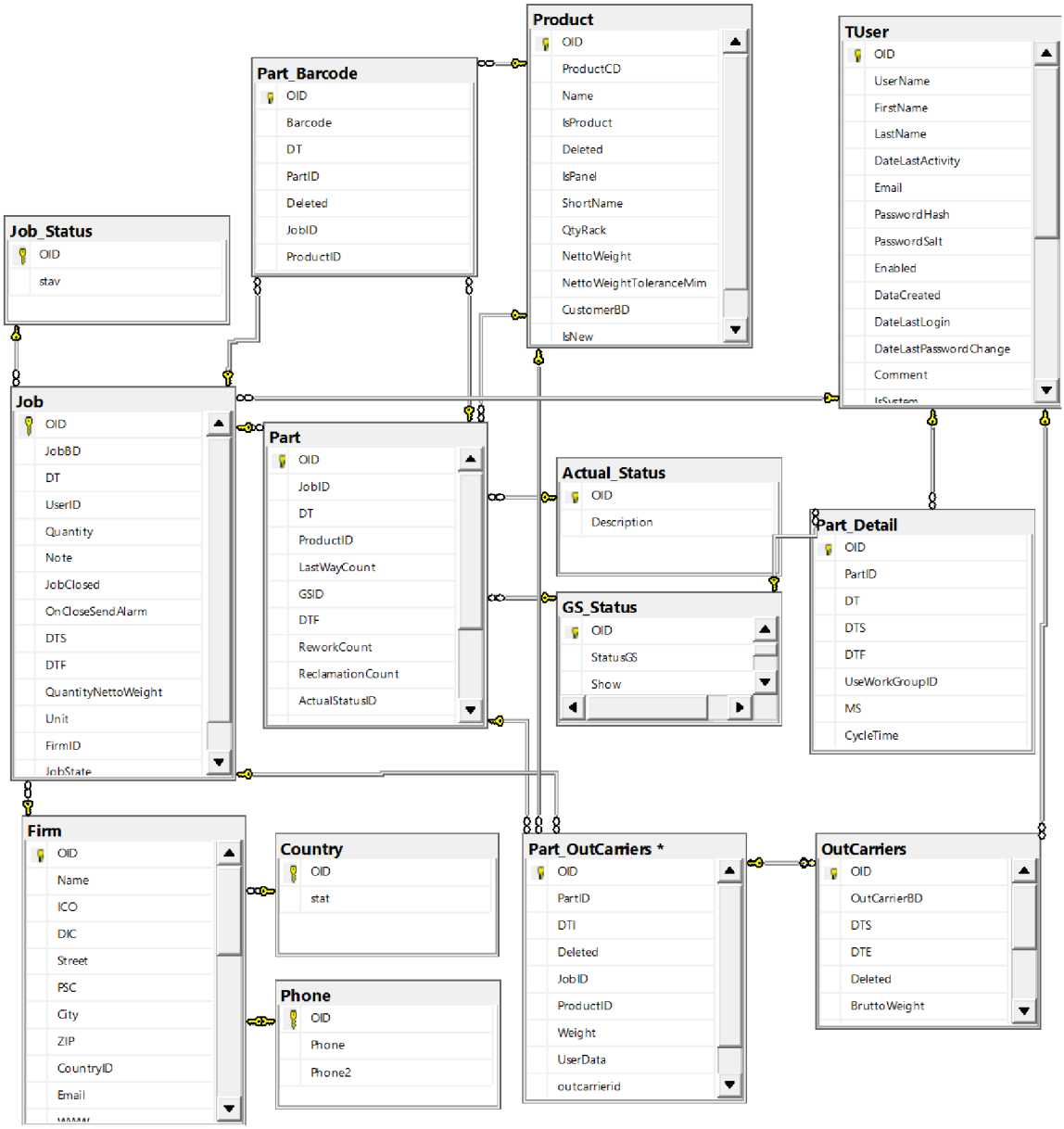
Obrázek č. 49: Tabulka Firm (Zdroj: vlastní zpracování)

3.2.4 Relace databáze

Tabulka	Vztahy relací
Job - Part_Barcode	Binární 1:*, rodičovská tabulka je Job a dceřinná Part_Barcode
Job - TUser	Binární 1:*, rodičovská tabulka je TUser a dceřinná Job
Job - Firm	Binární 1:*, rodičovská tabulka je Firm a dceřinná Job
Currency - Firm	Binární 1:*, rodičovská tabulka je Currency a dceřinná Job
Country - Firm	Binární 1:*, rodičovská tabulka je Country a dceřinná Firm
Job - Part	Binární 1:*, rodičovská tabulka je Part a dceřinná Job
Job - Part_OutCarriers	Binární 1:*, rodičovská tabulka je Job a dceřinná Part_OutCarriers
Part - Part_OutCarriers	Binární 1:*, rodičovská tabulka je Job a dceřinná Part_OutCarriers
Part_Barcode - Part	Binární 1:*, rodičovská tabulka je Part a dceřinná Part_Barcode
Product - Part_Barcode	Binární 1:*, rodičovská tabulka je Product a dceřinná Part_Barcode
Part - Actual_Status	Binární 1:*, rodičovská tabulka je Acutal_Status a dceřinná Part
GS_Status - Part	Binární 1:*, rodičovská tabulka je GS_Status a dceřinná Part
Part - Product	Binární 1:*, rodičovská tabulka je Product a dceřinná Part
Part_OutCarrier - Product	Binární 1:*, rodičovská tabulka je Product a dceřinná Part_OutCarriers
Part_OutCarrier - OutCarriers	Binární 1:*, rodičovská tabulka je Part_OutCarriers a dceřinnáOutCarriers
GS_Status - Part_Detail	Binární 1:*, rodičovská tabulka je GS_Status a dceřinná Part_Detail
OutCarriers - TUser	Binární 1:*, rodičovská tabulka je TUser a dceřinná OutCarriers
Part_Detail - TUser	Binární 1:*, rodičovská tabulka je TUser a dceřinná Part_Detail
Job_Status - Job	Binární 1:*, rodičovská tabulka je Job_Status a dceřinná Job
Firm - Phone	Binární 1:1 s povinnou participaci na jedné straně, rodičovská tabulka je Firm a dceřinná Phone

Obrázek č. 50: Relace (Zdroj: vlastní zpracování)

3.2.5 ER Diagram databáze



Obrázek č. 51: ER diagram (Zdroj: vlastní zpracování)

3.3 Business logika

Společnost využívá uložené procedury v SQL i pro složitější procedury s business logikou. Procedury s business logikou by měly být programovány v programovacím jazyce, který je k tomu lépe uzpůsoben.

Tento problém by bylo velmi obtížné opravit, protože databáze je obsáhlá a obsahuje mnoho skriptů. S vedením společnosti jsem o tom vedl diskuzi. Vedení má v budoucnu v plánu, vytvořit novou verzi svého softwaru xTrace, kde zajistí, že se budou psát složitější procedury s business logikou v příslušném programovacím jazyce.

3.4 Odstranění rizik z posouzení zranitelnosti

Zde musíme projít všechny bezpečností rizika a po zhodnocení usoudit, jestli se jedná o opravdové riziko a následně ho vyřešit, nebo ne a je to v pořádku.

3.4.1 Riziko VA2108 role

Server SQL poskytuje role, které umožňují spravovat oprávnění. Role jsou bezpečnostní principy, které seskupují jiné principy. Role na úrovni databáze mají rozsah oprávnění pro celou databázi. V tomto případě schválíme toto riziko jako výchozí, protože všechny role jsou správně nastavené programátorům ve firmě.

3.4.2 Riziko VA1102 důvěryhodný bit

Vlastnost důvěryhodné databáze slouží k určení, zda instance serveru SQL důvěřuje databázi a jejímu obsahu. Pokud je tato možnost povolena, mohou databázové moduly (například uživatelsky definované funkce nebo uložené procedury), které používají kontext impresonování, přistupovat ke zdrojům mimo databázi. Důvěryhodný bit je mechanismus řízení přístupu, který umožňuje funkce, které mohou vést k vyhodnocení privilegií, jako je CLR a impersonation server-scope. V databázi je povolena CLR a programátoři musí mít přístup k dalším databázím, proto schválíme toto riziko jako výchozí.

3.4.3 Riziko VA1256 CLR

Sestavy CLR lze použít ke spuštění libovolného kódu Microsoft .NET frameworku na serveru SQL. Špatné využití sestav CLR může přinést bezpečnostní závadu pro instanci SQL serveru a všechny ostatní síťové prostředky, které jsou z ní přístupné. Ale v naší situaci je CLR chráněno příslušnými oprávněními, díky tomu schválíme toto riziko jako výchozí.

3.4.4 Riziko VA1095 oprávnění veřejných uživatelů

Každé přihlášení k serveru SQL patří do role veřejného serveru. Pokud hlavnímu uživateli serveru nebyla udělena nebo odepřena konkrétní oprávnění k zabezpečenému objektu, uživatel zdědí oprávnění udělená veřejnému uživateli k tomuto objektu. Udělování oprávnění zadavatelům prostřednictvím výchozí role public tomuto účelu odporuje.

```
REVOKE EXECUTE ON TYPE::[DummyPanelItemTestResult] FROM PUBLIC
REVOKE EXECUTE ON TYPE::[IDList] FROM PUBLIC
REVOKE EXECUTE ON TYPE::[IDDictionary] FROM PUBLIC
REVOKE EXECUTE ON TYPE::[IDStringDictionary] FROM PUBLIC
REVOKE EXECUTE ON TYPE::[RPPartIDList] FROM PUBLIC
REVOKE EXECUTE ON TYPE::[PartDetailList] FROM PUBLIC
```

3.4.5 Riziko VA1219 Transparentní šifrování dat

Transparentní šifrování dat pomáhá chránit databázové soubory před vyzrazením informací tím, že provádí šifrování a dešifrování databáze, souvisejících záloh a souborů protokolu transakcí v reálném čase, aniž by vyžadovalo změny v aplikaci. Transparentní šifrování dat chrání data, což znamená, že data a soubory protokolu jsou šifrovány při ukládání na disk.

První musíme vytvořit hlavní klíč v hlavní databázi.

```
USE Master
go
create master key encryption
BY Password= <Heslo>
GO
```

Dalším krokem je vytvoření certifikátu chráněného hlavním klíčem.

```
CREATE CERTIFICATE TDE
WITH
SUBJECT='Database_Encryption';
GO
```

Nyní vybereme provozní databázi a vytvoříme spojení mezi certifikátem a databází. Dále zvolíme typ šifrovacího algoritmu AES_256.

```
USE Provozni_DB
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TDE;
GO
```

V posledním kroku v databázi povolíme šifrování. Šifrování databáze může trvat déle v závislosti na velikosti databáze.

```
ALTER DATABASE Provozni_DB
SET ENCRYPTION ON;
GO
```

Je také důležité vytvořit zálohu certifikátu v případě výpadku serveru a obnovy na bezpečném místě.

```
BACKUP CERTIFICATE TDE
TO FILE = 'C:\temp\TDE'
WITH PRIVATE KEY (file='C:\temp\TDE.pvk',
ENCRYPTION BY PASSWORD=<Heslo>
```

3.4.6 Riziko VA1244 osiřelí uživatelé

Uživatel databáze, který existuje v databázi, ale nemá odpovídající přihlášení v hlavní databázi nebo jako externí zdroj, se označuje jako osiřelý uživatel a měl by být buď odstraněn, nebo vytvořeno uživatelské přihlášení. Osiřelí uživatelé jsou obvykle známkou chybné konfigurace. Tito uživatelé představují riziko, protože potenciální útočníci k nim mohou získat přístup a zdědit jejich oprávnění v databázi.

```
DROP USER [excel]
```

3.4.7 Malá rizika

Zbývající malá rizika VA2109 a VA1054 označíme jako výchozí, protože nepředstavují žádnou hrozbu.

3.5 Zhodnocení návrhu

Cílem práce bylo zanalyzovat část produkční databáze a navrhnout zlepšení ve firmě. Pomocí naindexování sloupců v tabulkách se vyřešily problémy s uváznutím a zlepšila se rychlost spojování a dotazování v tabulkách. Dále se optimalizovala tabulka Firm, která je nyní správně navržená a přehlednější. Díky úpravám databáze je bezpečnost významně zvýšená vůči odcizení dat jak zvenku, tak i zevnitř. Doporučil bych i pravidelné aktualizace.

Závěr

Cílem této práce bylo zanalyzovat současný stav části výrobní databáze a na základě analýzy navrhnout řešení nedostatků pro zlepšení stavu.

V teoretické části jsou popsány základní pojmy z oblasti databázových systémů, databázových modelů a jazyka SQL, které sloužily jako podklad pro zbývající části práce.

Druhá kapitola se zabývá analýzou současného stavu databáze. Zde jsou popsány základní informace o společnosti, informační technologie, výrobní proces a databáze.

Třetí kapitola obsahuje návrh řešení na základě analýzy současného stavu. V této části byly navrhnuty a následně implementovány změny pro zlepšení stavu databáze bez dodatečných nákladů.

Seznam použitých zdrojů

[1] KOCH, Miloš a Bernard NEUWIRTH. Datové a funkční modelování. Vyd. 4., rozšířené. Brno: Akademické nakladatelství CERM, 2010, 142 s. : il., grafy, tab. ISBN 978-80-214-4125-5.

[2] KŘÍŽ, Jiří a Petr DOSTÁL. Databázové systémy. Brno: Akademické nakladatelství CERM, 2005, 111 s. : il. ISBN 80-214-3064-8.

[3] POKORNÝ, Jaroslav a Michal VALENTA. Databázové systémy. 2. přepracované vydání. Praha: Česká technika - nakladatelství ČVUT, 2020, 293 stran : ilustrace. ISBN 978-80-01-06696-6.

[4] CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. Mistrovství – databáze: profesionální průvodce tvorbou efektivních databází. Brno: Computer Press, 2009, 584 s. : il. ISBN 978-80-251-2328-7.

[5] BARTÍK, V. Datové sklady – DS [online]. Vysoké učení technické v Brně, [cit. 2022-05-01].

[6] Kříž, Jiří. Relační databáze – RD [online]. Vysoké učení technické v Brně, [cit. 2022-05-01].

[7] Transaction control [online]. [cit. 2022-01-02]. Dostupné z: https://www.ibm.com/docs/en/psfa/7.2.1?topic=categories-transaction-control.

[8] Data Control Language [online]. [cit. 2022-01-02]. Dostupné z: https://www.ibm.com/docs/en/psfa/7.2.1?topic=categories-data-control-language.

[9] Diagram data objects [online]. [cit. 2022-01-02]. Dostupné z: https://www.ibm.com/docs/en/informix-servers/14.10?topic=model-diagram-data-objects.

[10] STEPHENS, Ryan K, Ronald R PLEW, Arie JONES a Lukáš KREJČÍ. Naučte se SQL za 28 dní. Brno: Computer Press, 2010, 728 s. : il. ISBN 978-80-251-2700-1.

[11] SQL database functions [online]. [cit.2022-01-02]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/functions/functions?view=sql-server-ver15.

[12] Vulnerability assessment for SQL Server [cit.2022-01-02]. Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-vulnerability-assessment?view=sql-server-ver15.

[13] STEPHENS, Ryan K, Ronald R PLEW, Arie JONES a Lukáš KREJČÍ. Naučte se SQL za 28 dní. Brno: Computer Press, 2010, 728 s. : il. ISBN 978-80-251-2700-1.

[14] KOCH, Miloš a Viktor ONDRÁK. Informační systémy a technologie. Vyd. 3. Brno: Akademické nakladatelství CERM, 2008, 166 s. : il., grafy, tab. ISBN 978-80-214-3732-6.

Seznam použitých obrázků

Seznam použitých tabulek