VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH GRAFOVÉ DATABÁZE NA ZÁKLADĚ RELAČNÍ DATABÁZE
DESIGN OF GRAPH DATABASE BASED ON RELATIONAL DATABASE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jiří Schwertschal**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Luhan, Ph.D., MSc**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Jiří Schwertschal**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Vedoucí práce:	**Ing. Jan Luhan, Ph.D., MSc**
Akademický rok:	2017/18

Ředitel ústavu 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:

Návrh grafové databáze na základě relační databáze

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 této práce je návrh a následná implementace grafové databáze na základě relační databáze, sloužící jako rezervační systém konkrétního subjektu. Přičemž návrh musí plně reflektovat na požadavky subjektu. Pro vlastní realizaci bude využito platforem Microsoft SQL Server a Apache Tinkerpop.

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.

GARCIA-MOLINA, H., J. D. ULLMAN and J. WIDOM. Database systems: The Complete Book. 2nd ed. Upper Saddle River, N.J.: Pearson Prentice Hall, 2008. 1248 p. ISBN 978-0131873254.

HARRINGTON, J. L. Relational database design and implementation. 4th edition. Cambridge, MA: Elsevier, 2016. 712 p. ISBN 978-0128043998.

OPPEL, A. Databáze bez předchozích znalostí: Průvodce pro samouky. Brno: Computer Press, 2006. 320 s. ISBN 80-251-1199-7.

ROBINSON, I., J. WEBBER a E. EIFREM. Graph databases: New Opportunities for Connected Data. 2nd ed. Beijing: O'Reilly Media, 2015. 238 p. ISBN 978-1491930892.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2017/18

V Brně dne 28.2.2018

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

Abstrakt

Obsahem mé bakalářské práce je návrh převedení SQL databáze určené pro rezervační systém do databáze grafové. Práce je rozdělena na teoretickou část, ve které jsem uvedl informace potřebné pro vytvoření návrhu. V druhé části jsem provedl analýzu současného stavu. A nakonec v poslední části se nachází vlastní řešení.

Abstract

The content of my bachelor thesis is the design of converting SQL database meant for reservating systém to the graph database. The thesis is divided into theoretical part, where I provided information needed to create the design. In the second part I made analysis of the current situation. And in the end in the last part is designed solution.

Klíčové slova

SQL, primární klíč, databáze, dotaz, Cypher, graf, uzel, vztah, vlastnost, Neo4j

Key words

SQL, primary key, database, query, Cypher, graph, node, relationship, property, Neo4j

Bibliografická citace

SCHWERTSCHAL, J. *Návrh grafové databáze na základě relační databáze*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2018. 70 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 ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 17. května 2018 ……

Jiří Schwertschal

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce Ing. Janu Luhanovi, Ph.D., MSc. za všechny rady a pomoc, které poskytl. Také bych chtěl poděkovat svému oponentovi, rodině a přátelům.

OBSAH

ÚVOD

Data jsou jedním z největších trendů doby. Dotýkají se nás všech všemi možnými způsoby. Každý člověk má dnes data jeho se týkající, ať už jde o jméno a příjmení, práce, kterou vykonal nebo výplata kterou si prací vydělal. Ve velkých společnostech je velké množství lidí a každého z oněch lidí se týkají určitá data. S tímto velkým množstvím dat se musí společnosti naučit pracovat, aby by bylo efektivně využito. K tomu slouží databáze.

Tato bakalářská práce slouží jako návrh pro převedení staré nedostačující relační databáze do databáze grafové, jež zvládá pracovat s velkým množstvím dat efektivněji.

1 CÍL A METODIKA PRÁCE

Cílem této bakalářské práce bude převedení relační databáze, sloužící pro rezervační systém, do databáze grafové. Databáze musí splňovat všechny požadavky majitele a zároveň napravit nedostatky databáze předchozí.

Návrh budu realizovat za pomoci Microsoft SQL serveru 2016 a Neo4j databáze. Vytvořit cílovou databázi v Neo4j jsem se rozhodl z několika důvodů. Prvním je stabilita. Neo4j databáze jako první používala tzv. property graph. Je v současné době známa jako jedna z nejstabilnějších grafových databází jak při čtení, tak i psaní.

Dalším důvodem je velká komunita díky open source licence a snadné dohledání informací potřebných pro vytvoření databáze např. v dokumentaci Neo4j.

Dále díky bezindexovosti se zkracuje čas čtení. Je lehce naučitelná a lehce využitelná. Podporuje velké množství programovacích jazyků a neposlední v řadě podporuje i jiné dotazovací jazyky například gremlin od Tinkerpop.

První část práce se bude věnovat teoretickým východiskám, které vysvětlí základní pojmy a problematiku v oblastech výše zmíněných, tedy databázové systémy, relační databáze a databáze grafové.

V části druhé se pak bude nacházet analýza současného stavu, tedy stavu již existující relační databáze. Bude obsahovat nejen informace o databázi, ale i její výhody, nevýhody a důvody nahrazení z nich vyplývající.

Ve třetí části pak uvedu vlastní návrh řešení. Bude zohledněna předchozí analýza a následně jí bude přizpůsoben samotný návrh.

2 TEORETICKÁ VÝCHODISKA PRÁCE

Před vytvořením samotného návrhu musím uvést a vysvětlit základní pojmy a principy potřebné při vytváření databázových systémů jak relačních, tak grafových. Tyto pojmy pak budu nadále využívat při popisu mého řešení problematiky. V první části hierarchicky definuji pojmy jako jsou data, databáze, databázový model.

V dalších částech definuji pojmy týkající se přímo relační databáze. Nejdříve jednotlivé části jako jsou entity, atributy, databázové tabulky a datové typy. Dále pak relace a jaké druhy relací existují, primární a cizí klíče které je tvoří, a nakonec uvedu základní klauzule jazyka SQL, pomocí něhož databázi budu tvořit.

V třetí části poté budu popisovat pojmy grafové databáze jež pak využijeme k tvorbě návrhu. Vysvětlím princip grafu, grafové databáze. Dále popíši, jak se tvoří samotné dotazy, a nakonec popíši využitý dotazovací jazyk Cypher a jeho hlavní klauzule.

Na konci kapitoly bude shrnutí a důvody využití grafové databáze.

2.1 Základní pojmy

Tato kapitola se zabývá nejzákladnějšími pojmy týkajících se dat, databází a relačních databází. Také zde vysvětlím, jaké jsou nejdůležitější části databáze.

2.1.1 Data a informace

„Data můžeme chápat jako surová (nezpracovaná) fakta, která mají určitou důležitost pro jednotlivce nebo skupinu.“ (3, str. 8)

„Informace jsou data, která prošla zpracováním, nebo dostala strukturu, která jim dává pro jednotlivce nebo organizaci význam.“ (3, str. 8)

„Primárním cílem databázových systémů je podávání smysluplných informací. Smysluplnou informaci však jsme schopni získat pouze za předpokladu, že v databázi existují smysluplná data. Vztah mezi daty a informacemi si můžeme pro snazší zapamatování zjednodušeně vyjádřit: Data se ukládají do databáze a z databáze získáváme informace.“ (3, str. 8)

2.1.2 Databázový systém a databáze

Databázový systém vzniká sloučením dat a nástrojů pro práci s daty. Každý databázový systém musí tedy obsahovat důležité nástroje, jako jsou pro vytvoření a úpravu dat, struktury dat, zajištění fyzické a logické nezávislosti dat a případně zabezpečení a zálohování dat (2).

Fyzická nezávislost dat v podstatě znamená uložení dat na disku odděleně od způsobu práce s nimi. Logická zase když změna logické struktury nezmění nijak existující programy a dotazy pracující s daty (2).

Databáze je v soubor vzájemně souvisejících dat, se kterými pracujeme jako s jedním celkem. Samozřejmě jde jen o velmi obecný popis, protože jednotlivými produkty je velké množství variant a druhů (1).

Databázový objekt je datová struktura uložená v databázi, jejíž typy se liší podle výrobce a databázového modelu. Databázový model nám udává způsob, jakým způsobem jsou data uspořádána v databázi. Soubor jsou příbuzné záznamy uložené v jedné struktuře (1).

„Databázový model (model databází) je způsob uspořádání dat v databázi, která tak odráží podobu reálného světa.“ (1)

2.2 Relační databáze

V této části budou vysvětleny pojmy týkající se relačních databází použitých v této práci.

2.2.1 Entity

„Je osoba, místo, věc, událost nebo myšlenka, o níž shromažďujeme nějaká data. Jinými slovy jsou entity „předměty“ z reálného světa, které jsou pro nás dostatečně zajímavé, takže o nich sledujeme určité údaje a zaznamenáváme je do databáze. Entitu reprezentuje v diagramu obdélník.“ (1, str. 40)

Jako příklad z naší databáze může sloužit například „zaměstnanec“.

2.2.2 Atributy

„Atribut je jednotka faktů, která entitu nějakým způsobem charakterizuje nebo popisuje. V diagramu s konceptuálním návrhem znázorňujeme atributy jako názvy uvnitř obdélníku.“ (1, str. 41)

„Řekli jsme si, že atribut je jednotkou faktů o entitě – každý atribut by totiž měl být atomický, tedy takový, aby jeho hodnotu nebylo možné smysluplně dále rozdělit do několika menších jednotek. Jinými slovy atribut je nejmenší pojmenovaná jednotka dat definovaná v databázovém systému.“ (1, str. 41)

Ovšem existují výjimky, kdy například uvedení celé adresy do jednoho atributu, jenž by se dala rozdělit na menší, usnadní tisk adresních štítků (1).

Příkladem atributu ve výše zmíněné entitě může být například křestní jméno, příjmení nebo telefon.

2.2.3 Databázové tabulky

V databázovém prostředí pracujeme hlavně s databázovými tabulkami. Jedná se jen o jednoduché tabulky s jedním titulkovým řádem a bez titulků u řádků. Kromě titulního řádku jsou všechny data uživatelská. Základními prvky jsou sloupec, řádek a hodnota (2).

Tabulka obsahuje **sloupce** (atributy) a **řádky** (záznamů) obsahujících data. Každá tabulka musí obsahovat aspoň jeden sloupec a nemusí být žádný řádek, což znamená, že tabulka je bez dat. Data jsou hodnoty zadané uživatelem do buněk, které jsou průsečíkem určitého řádku a sloupce a tvoří mezi nimi vztah (2).

„Každému sloupci musíme přiřadit jedinečný název (tedy jedinečný v rámci tabulky) a datový typ. Datový typ je přitom jakási kategorie formátu konkrétního sloupce.“ (1, str. 47)

Definice datového typu má několik výhod jako omezení množiny povolených dat na takové znaky, které mají pro daný datový typ smysl (například datum). Dále pak napomáhá systému v efektivním ukládání dat. Například u číselných dotavých typů je poté provádět určité matematické operace, kdežto u čísla uloženém jako text tyto možnosti ztrácíme (1)

Příklad databázové tabulky můžeme vidět v tabulce č. 1.

Tabulka č. 1: Příklad tabulky (vlastní)

Křestní jméno	Příjmení	Telefon
Jiří	Novák	412 125 456
Karel	Brabec	789 456 123

2.2.4 Základní datové typy

V předchozí kapitole bylo vysvětleno, co jsou datové typy zde uvedu pár základních, které jsou používány v Microsoft SQL Server, jenž budeme používat v naší práci.

Typy dělíme na znakové, celočíselné, desítkové číselné a časové. Znakové pak dělíme na znakové s pevnou délkou a s délkou proměnnou: Příkaz CHAR(30) uloží slovo ‚Karel' jako Karel společně s dalšími dvaceti pěti mezerami, zatímco VARCHAR(30) tyto mezery při vyvolání od řetězce usekne (1).

Tabulka č. 2: Tabulka datových typů (7, vlastní překlad)

Datový typ	MS SQL	Popis
Znakový s pevnou délkou	CHAR(*n*)	Řetězec s pevně danou délkou s počtem charakterů (*n*)
Znaková s proměnnou délkou	VARCHAR(*n*)	Řetězec s proměnlivou délkou s maximálním počtem charakterů (*n*)
Celočíselný	INTEGER nebo INT	Ukládá celočíselné hodnoty od -2^31 do 2^31
Celočíselný	TINYINT	Ukládá celočíselné hodnoty od 0 do 255
Desítkový číselný	FLOAT	Ukládá desítkové číselné hodnoty
Časový	DATE	Ukládá hodnoty data
Časový	TIME	Ukládá hodnoty času

2.2.5 Relace

„Vztahy (dále již jako relace) popisují vzájemné vztahy neboli „asociace“ mezi entitami.“ (1, str. 41)

Jelikož do databází vkládáme data, která jsou na sebe určitým způsobem navázána. Potřebujeme tyto vazby nějak reprodukovat do databáze též a k tomu slouží právě relace. (1).

„V diagramu konceptuálního návrhu jsou relace naznačeny jako čáry propojující jedno nebo více entit. Na každém konci relační čáry je dále znázorněna maximální kardinalita vztahu, tedy největší počet instancí jedné entity, které mohou být sdruženy s entitou na opačném konci.“ (1, str. 41)

Každá kardinalita má své označení maximální a minimálního počtu instancí a má svůj vlastní symbol na relační čáře. Maximální kardinalita může nabývat hodnoty jedna (tehdy čára nemá žádný symbol) nebo více (na konci čáry je rozvětvení). Před tímto symbolem se pak nachází druhý, který udává minimální kardinalitu relace, tedy nejmenší počet instancí entity. Ta může nabývat hodnotu nula (na čáře je zakreslen kroužek) nebo hodnotu jedna (čára je přetnuta krátkou kolmou čárkou) (1).

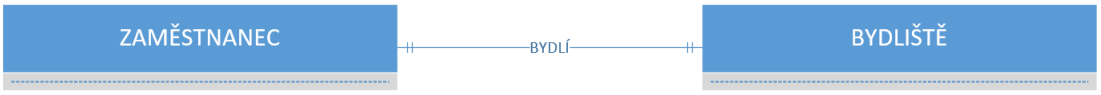
Příklady symbolů kardinality můžeme nalézt na obrázku č.1. Kde na levé straně relace je symbol jedna a právě jedna (dvě kolmé čárky) a na straně pravé je nula a více (kroužek a rozvětvení).

Obrázek č. 1: Kardinalita relace (vlastní)

Relace můžeme dělit do tří typů: *jedna k jedné, jedna k více a více k více*. **Relace jedna k jedné** je taková, kdy k hodnotě v jedné tabulce můžeme přiřadit právě jednu v druhé

tabulce. Tato relace je velice vzácná, protože v praxi takováto relace znamená jasnou chybu návrhu, kdy obě entity můžeme sloučit do jedné (1).

Jako příklad můžu použít rozdělení atributů tabulky zaměstnanec kdy by právě jeden zaměstnanec měl právě jedno bydliště jako na obrázku č.2.



Obrázek č. 2: Relace 1:1 (vlastní)

Relace jedna k více znamená, že hodnota první tabulky může být přiřazena k jedné nebo více hodnotám v druhé tabulce. Je to velice běžná relace a v podstatě jednou základní relací k tvorbě databází (1).

Jako příklad v naší databázi máme případ kdy jeden a více zaměstnanců spadá pod právě jeden department jako na obrázku č.3.

Obrázek č. 3: Relace M:1 (vlastní)

Další neméně důležitou je **relace více k více**. Mezi dvěma tabulkami je jedna nebo více návazností mezi hodnotami. V praxi se tato relace vyskytuje velice často, avšak neexistuje její přesná realizace, proto se musí řešit pomocí dekompozice. Samotnou relaci není v praxi možné fyzicky vytvořit, proto si pomáháme dekompozicí na dvě vazby vybraných z typů výše zmíněných (1).

Příkladem v naší databázi může být relace zaměstnanec : místnost uvedená na obrázku č.4. Kde je relace kdy jeden a více zaměstnanců může rezervovat jednu a více místností.

Obrázek č. 4: Relace M:N (vlastní)

Poslední relací je **rekurzivní relace**. Předtím byly zmíněny relace mezi entitami dvou různých typů. Relace však se může navazovat mezi entitami stejného typu. To jsou právě rekurzivní relace. Například kdyby z nějakého důvodu bylo potřeba zaznamenat manželství mezi dvěma zaměstnanci, nebo který zaměstnanec je nadřízen kterému.

2.2.6 Primární, kandidátní a cizí klíč

Primární klíč je unikátní atribut, který identifikuje každý řádek v tabulce. Je to důležité kvůli možnosti získat jakýkoliv kousek z dat, které vložíme do databáze (5, překlad vlastní).

K získání jakékoliv informace z databáze potřebujeme přesně tři potřebné informace: název tabulky, název sloupce, a právě zmíněný primární klíč řádku ve kterém se žádaná informace nachází. Jestliže primární klíče jsou unikátní, můžeme si být naprosto jistí, že dostaneme ve výsledku ten řádek, jaký požadujeme (5, překlad vlastní).

Společně s unikátností, primární klíč nesmí nikdy obsahovat prázdnou hodnotu *null*. Null je speciální databázová hodnota znamenající „unknown.“ Neznamená to samé, co klasická hodnota nula nebo prázdná hodnota (5, překlad vlastní).

Při výběru primárního klíče je možné narazit na několik potenciálních primárních klíčů. Ty nazýváme **kandidátními klíči**. Z nich si pak vybereme jeden primární klíč a zbylé jsou tzv. alternativní klíče (5, překlad vlastní).

„Primární klíče se používají v dalších tabulkách pro určení vazeb mezi tabulkami. U každé výplaty v tabulce výplat musíme určit, kterému zaměstnanci byla vyplacena. V tabulce zaměstnanci podobně určujeme, do kterého oddělení zaměstnanec patří. Jako odkaz použijeme v tabulce výplat rodné číslo zaměstnance a v tabulce zaměstnanci číslo oddělení. V obou případech se jedná o primární klíče v odkazovaných tabulkách, protože právě u primárních klíčů máme jistotu, že jednoznačně identifikují jeden řádek v odkazované tabulce. Použití primárního klíče cizí tabulky pro vytvoření odkazu se nazývá cizí klíč. Cizí klíč jako zprostředkování vazby do jiné tabulky se bude vždy skládat ze stejných sloupců jako primární klíč v odkazované tabulce. V případě, že bychom potřebovali vytvořit odkaz do tabulky, ve které existuje složený primární klíč, museli bychom přidat tolik sloupců, kolik je v odkazované tabulce sloupců v primárním klíči.“ (2, str. 22)

2.2.7 SQL

„SQL (Structured Query Language) je standartní jazyk, který se používá při komunikaci s relačními databázemi. … Dotaz (query) je požadavek, který se odesílá databázi. Na základě tohoto požadavku databáze žadateli zpětně poskytne určitou odpověď.“ (4, str. 33)

„SQL se řadí mezi neprocedurální neboli deklarativní jazyky. To znamená, že počítači sdělíte, jaké výsledky požadujete. Přitom nemusíte definovat, jak ty to výsledky získat.“ (4, str. 33)

Sdělit počítači tyto výsledky je možné třemi způsoby, které záleží na tom, jakou technologii používáme. Jde o klienty příkazového řádku, grafické a webové klienty (4).

Jazyk SQL se řídí syntaktickými konvencemi, díky nimž pak tvoříme příkazy (4).

Uplatňují se například tyto základní konvence:

- Každý příkaz začíná klíčovými slovy. Například slovo SELECT (4).
- Každý příkaz končí oddělovačem, což je většinou středník (4).
- Struktura příkazů se podobá anglickým větám (4).
- Příkazy tvoří řadu klíčových slov, které většinou dodržují určité pořadí (4).
- Prvky lze zapisovat velkými, malými písmeny nebo jejich kombinacemi (4).
- Jednotlivé atributy nebo data se oddělují čárkami (4).

2.2.8 Základní klauzule jazyka SQL

Klauzule jazyka SQL jsou členěny do několika skupiny pro nás jsou však důležité skupiny pro manipulaci s daty a pro vytváření a rušení tabulek. Manipulaci s daty rozumíme vyhledávání dat, přidávání nových řádků do tabulek, aktualizaci hodnot v řádcích anebo mazání existujících řádků (2).

Select

„Pro vyhledávání dat podle různých kritérií je v jazyku SQL k dispozici pouze příkaz SELECT. Protože dotazy do databáze mohou být různorodé, má i příkaz SELECT mnoho variant.“ (2, str. 33)

Nejzákladnější varianta je ta, s jíž pomocí jsme schopni získat základní informace z databáze (6).

Základní syntaxe dotazu je ve tvaru: SELECT (názvy sloupců) FROM (název tabulky). Dále pak je možné do dotazu přidat podmínky pomocí klauzule WHERE, které jsou pak vyjádřeny určitým operátorem jako je například „=“ operátor (6).

Příklad SELECT:

```
SELECT Jmeno, Prijmeni
FROM Zamestnanec
WHERE Jmeno = ‚Karel‘;
```

Create table

Klauzule CREATE TABLE umožňuje vytvořit databázovou tabulku. Při jejím vytváření je potřeba napsat název a názvy sloupců, které bude tabulka obsahovat. U každého sloupce musí být i zadaný datový typ hodnot (2).

Příklad CREATE TABLE:

```
CREATE TABLE Zamestnanec (
id_zam int NOT NULL PRIMARY KEY,
jmeno varchar(20) NOT NULL,
telefon int);
```

Drop table

Pomocí této klauzule je možno smazat existující tabulku včetně všech řádků. Stačí jen napsat např. **DROP TABLE Zaměstnanec.** Po odstranění tabulky tímto způsobem není možné tabulku obnovit (2).

Insert

Po vytvoření tabulky je potřeba ji naplnit daty. K tomu slouží klauzule INSERT. Syntaxe dotazu vypadá následovně: INSERT INTO (název tabulky) VALUES (seznam dat rovnající se počtu sloupců). Případně je možno místo nějakého data vložit hodnotu *null* (2).

Příklad INSERT:

```
INSERT INTO Zamestnanec
VALUES (‚Karel‘, ‚Novák‘, 123 456 789);
```

Update

Občas při práci s databází je možné, že potřebujeme upravit data v tabulce. K tomuto účelu slouží klauzule UPDATE. Při jeho používání se nemění počet řádků tabulky. Nejjednodušší formou klauzule se dá upravit jeden sloupec u všech řádků tabulky. Při detailnější úpravě je potřeba používat podmínky WHERE. V dotazu níže zvyšujeme plat zaměstnanci Novákovi o 20 % (2).

Příklad UPDATE:

```
UPDATE Zamesnanec
SET plat = plat + 20/100
WHERE Prijmeni = ‚Novak‘
```

2.3 Grafové databáze

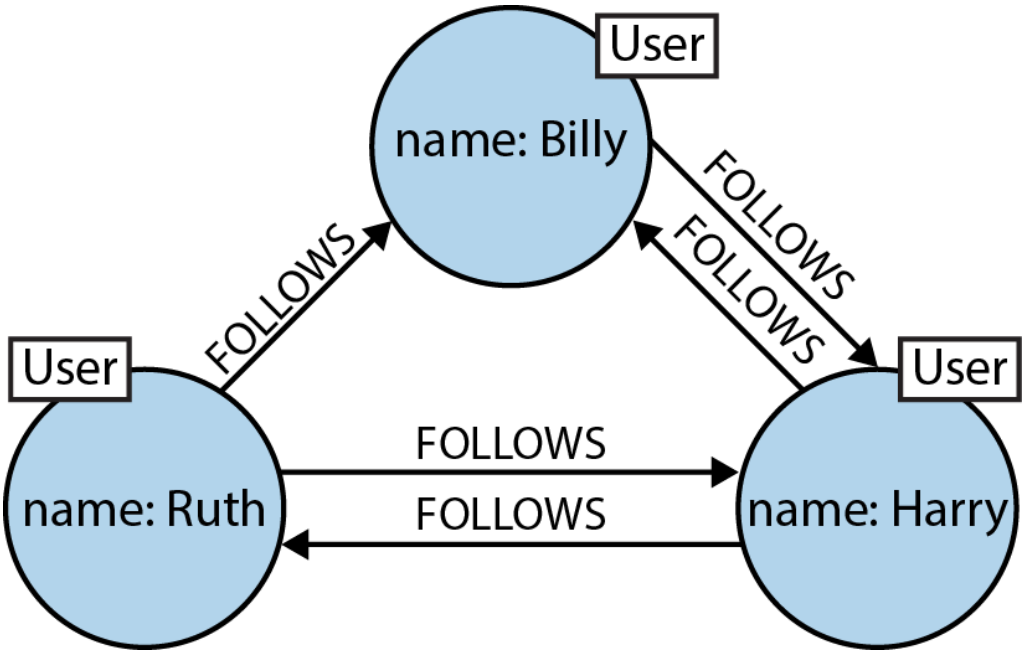
V této části se budu zabývat pojmy týkajících se grafu a grafových databází. Grafové databáze jako takové dnes denně využíváme, aniž bychom to vůbec tušili. Technologii využívají obrovské internetové společnosti jako sociální sítě Facebook, Twitter, Instagram a jim podobní nebo například Google.

2.3.1 Graf grafové databáze

Co je to graf? Jednoduše řečeno graf je kolekce **hran (edges)** a **vrcholů (vertices)** nebo jinak řečeno množina **uzlů (nodes)** a **vztahů (relationships)** je spojujících. Při porovnání s již dříve vysvětlenými relačními databázemi uzly zastupují entity a vztahy zastupují relace a ty dohromady vytváří model reálných entit a relací (8, vlastní překlad).

Nyní si můžeme uvést příklad, na kterém tyto prvky najdeme. Jako příklad využijeme data ze sociální sítě Twitter, které se dají jednoduše v grafu prezentovat (8, vlastní překlad).

Na obrázku č. 5 vidíme malou síť uživatelů Twitteru. Každý uzel je označen značkou *User*, která ukazuje roli v dané síti. Těchto rolí v síti může být nespočet. Uzly jsou pak spojeny vztahy, které vytváří kontext. Přesně řečeno, že uživatel Billy sleduje Harryho a Harry následně sleduje Billyho. Stejně tomu je mezi Ruth a Harrym, kteří se navzájem také sledují. Jediný rozdíl v grafu je mezi Ruth a Billy, kde Billy zatím Ruth nezačal sledovat (8, vlastní překlad).



Obrázek č. 5: Malý sociální graf (8, str. 2)

Ovšem pravý graf Twitteru je několik set milionkrát větší než uvedený příklad. Dále by v grafu mohly být uzly prezentující přidané příspěvky Ruth na její zdi nebo zprávy poslané mezi uživateli (8, vlastní překlad).

V předchozích odstavcích byl představen nejpoužívanější forma grafového modelu tedy **labeled property graph** (8, vlastní překlad).

„Ten má tyto charakteristiky:

- *Obsahuje uzly a vztahy.*
- *Uzly obsahují vlastnosti (hodnotově klíčové páry).*
- *Uzly mohou být označeny jedním nebo více označeními.*
- *Vztahy jsou pojmenovány a směrovány, a vždy začínají a končí v uzlu.*
- *Vztahy také mohou obsahovat vlastnosti.“ (8, str. 4, vlastní překlad)*

2.3.2 Definice grafové databáze

„Graph database management system (dále jen grafová databáze) je online databázový management systém s metodami Create, Read, Update a Delete (CRUD), které tvoří grafový model dat. Grafové databáze jsou obecně stavěny pro užití transakčního systému (OLTP).“ (8, str. 5, vlastní překlad)

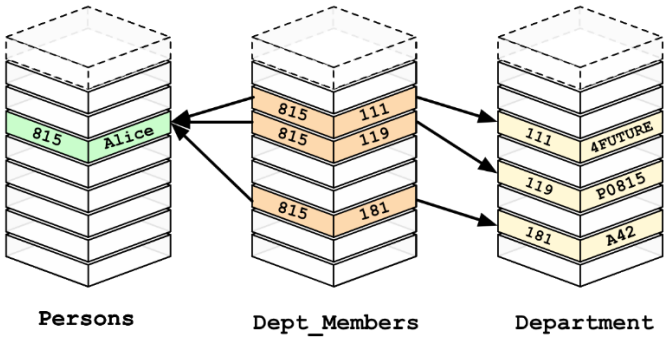
„Vztahy jsou v grafovém data modelu nejdůležitější částí, což je rozdíl oproti ostatním databázovým management systémům, kde relace tvoříme např. pomocí cizích klíčů. Sestavením jednoduchých abstrakcí z uzlů a vztahů v propojené struktury, grafové databáze nám dovolují postavit sofistikované modely podrobně mapující naši problematiku. Výsledné modely jsou jednodušší a zároveň výraznější než produkované tradiční relační databází.“ (8, str. 6, vlastní překlad)

2.3.3 Výhody grafových databází

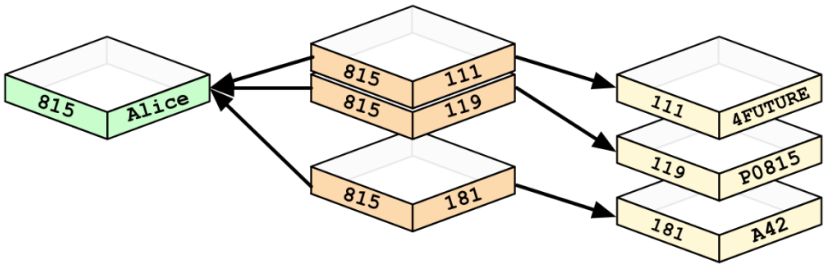
Grafové databáze nenabízí jen nový způsob modelování dat. To k rozšíření nové technologie nestačí. Je potřeba mít okamžité praktické benefity. V případě grafových databází tyto benefity jsou nám nabízeny ve formě velké množiny případů užití a datových vzorů, jichž výkon je vyšší o jeden či víc řádů, když jsou užity v grafu, a jichž latence je o hodně nižší v porovnání s ostatními typy databází. Kromě toho grafové databáze nabízí i velikou flexibilitu data modelu (8, vlastní překlad).

Jeden z hlavních důvodů využití grafové databáze je **výkon**, který je podstatně vyšší v porovnání s relačními a NOSQL databázemi. Kupříkladu u relačních databází tam, kde JOIN query výkon se zhoršuje, jak se zvyšuje množství dat, tam výkon grafové databáze zůstává konstantnější i když se množství dat nadále zvyšuje. Což znamená že ve výsledku čas provádění operací je ve výsledku při velkém objemu dat u relační databáze daleko vyšší (8, vlastní překlad).

Názorně je tento problém zobrazen na obrázcích č. 6 a č. 7, kde na obrázku č.6 můžeme vidět, jak se při vyhledání dat pomocí JOIN klauzule načte obrovské množství dat, zatímco na obrázku č. 7 je vidět, jak by množství dat vypadalo při použití grafové databáze.



Obrázek č. 6: Množství načtených dat relační databáze (11)



Obrázek č. 7: Množství načtených dat grafové databáze (11)

Druhým důvodem je vysoká flexibilita. Developeři a data architekti vyžadují, aby data byly spojována, jak potřeba určí, tedy mít možnost kdykoliv upravit strukturu či schéma. Grafy jsou velmi přizpůsobivé v tomto případě. Kdykoliv můžeme přidat nové typy vztahů, nové uzly či podgrafy do již existující struktury, aniž bychom narušili příkazy a funkcionality (8, vlastní překlad).

2.3.4 Jak grafové databáze fungují

„Na rozdíl od ostatních databázových systémů, jsou hlavní prioritou v grafových databázích vztahy. To znamená že aplikace nemusí nijak zasahovat do spojení dat používáním cizích klíčů nebo out-of-band procesy.

Existují dvě hlavní důležité vlastnosti grafových databází, kterým je třeba rozumět: ukládání grafů a grafový zpracovávací engine.

Některé grafové databáze využívají přirozené grafové ukládání, které je speciálně designováno pro ukládání a práci s grafy, zatímco jiné místo toho používají relační nebo objektově orientované databáze. Nepřirozené ukládání je často pomalejší než přirozené.

Přirozené grafové zpracovávání je nejvíce účinné ve zpracování dat v grafu, protože spojené uzly fyzicky ukazují na sebe navzájem v databázi. Zatímco v nepřirozené databázi zpracovávací engine používá jiné významy pro create, read, update a delete operace.“ (8, vlastní překlad)

2.3.5 Vytvoření grafové databáze

Do začátku potřebujeme k vytvoření modelu si potřebujeme říct příběh z pohledu čtenáře, který nám pomůže v identifikaci entit a vztahů (8, vlastní překlad).

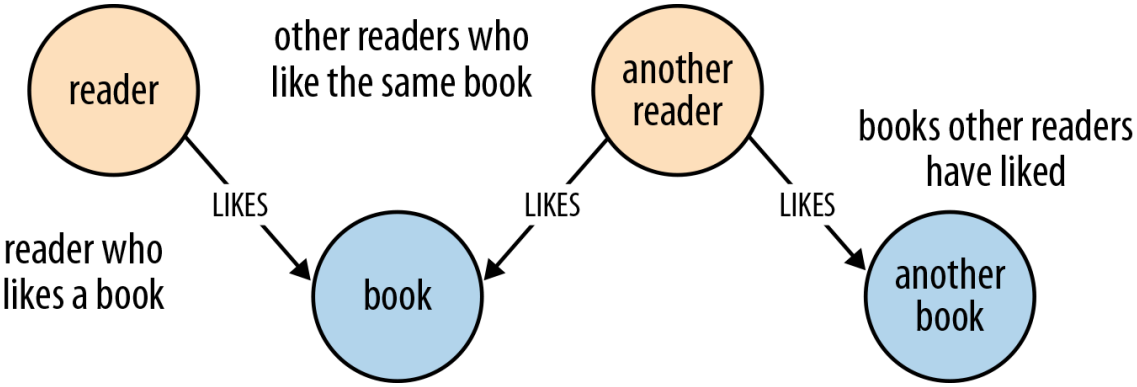
Tady je příklad takové věty:

***„JAKO** čtenář, co má rád určitou knihu, **CHCI** vědět, jaké jiné knihy měli čtenáři, kteří měli rádi stejnou knihou, mají rádi, **ABYCH** je mohl najít.“ (8, vlastní překlad)*

Z tohoto příběhu je jasné, co uživatel chce, a to nám pomůže vytvořit pohled na obsah dat. Z pohledu data modelování nám část „**JAKO“** udává dvě základní *entity*, které budou v grafu – ČTENÁŘ a KNIHA. K tomu nám udává první *vztah* MÁ RÁD, který je spojuje.

Část věty, která začíná na „**CHCI“** nám dává otázku potřebnou pro vytvoření modelu: jaké další knihy čtenáři, která měli rádi stejnou knihu, měli rádi? Tato otázka nám udává ten samý *vztah* MÁ RÁD a přidává k tomu dvě další *entity:* OSTATNÍ ČTENÁŘI a OSTATNÍ KNIHY (8, vlastní překlad).

Z těchto všech informací nám vzniká základní graf, jenž je na obrázku č. 8.



Obrázek č. 8: Datový model čtenářů a knih (8, str. 66)

Samozřejmě entity ostatní čtenáři a ostatní knihy jsou jen další entity čtenář a kniha, přesto je důležité si uvědomovat, že každý uzel představuje jednoho čtenáře.

Další věc, na kterou je nutno pamatovat je u technologie Neo4j. Tato technologie nepodporuje synchronní vztahy tedy vztahy, kde obrazně řečeno v naší databázi by nešlo se dotazovat na čtenáře, který má rád druhého čtenáře a zároveň druhý čtenář má rád prvního. Toto však u této technologie není potřeba je možné to obejít spojenými dotazy anebo předejít tomu již při tvorbě databáze (9).

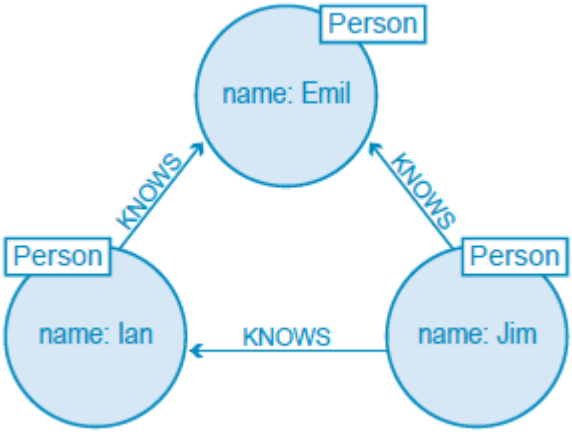
2.3.6 Jazyk Cypher

Zatímco relační database používají různé formy jazyka SQL, svět grafových databází je víc pestřejší. Mezi nejznámější jazyky patří například Gremlin nebo Cypher. V databázi, jenž budu používat, tedy Neo4j, je využíván jazyk Cypher (10, vlastní překlad).

Jazyk Cypher je hojně využívaný, open source a cross-platform query jazyk. Je vytvořen tak, aby byl jednoduše čten a pochopen developer, databázovými profesionály, ale i podnikatelskými subjekty (10, vlastní překlad).

Základní pojetí jazyku Cypher je, že dovolí uživateli se zeptat databáze na velmi specifická data. Dovoluje se zeptat na otázky typu "najdi věci jako tyto" a cestu jak se k těmto „věcem" dostat nakreslením cesty pomocí ASCII (10, vlastní překlad).

Pro vysvětlení si uvedeme příklad grafu na obrázku č. 9.



Obrázek č. 9: Příklad grafu (10, str. 15)

Když bychom chtěli přepsat tento základní graf do Cypher, napsali bychom:

```
(emil)<-[:KNOWS]-(jim)-[:KNOWS]->(ian)-[:KNOWS]->(emil)
```

Jak je možné vidět na uvedeném vzorci, Cypher svojí strukturou popisuje přirozenou cestu grafu. Tedy zobrazuje uzel **Jim** na, které jsou napojeny další dva uzly **Emil** a **Ian**. A spojením uzlů Ian a Emil vzniká trojúhelníkový graf (10, vlastní překlad).

Zobrazený vzorec sice popisuje jednoduše strukturu grafu, neodkazuje však na žádná data v něm. K tomu potřebujeme specifikovat hodnotové vlastnosti a štítky uzlů (10, vlastní překlad).

Zde je tedy vzorec k tomu určený:

```
(emil:Person {name:'Emil'})
<-[:KNOWS]-
(jim:Person {name:'Jim'})
-[:KNOWS]->
(ian:Person {name:'Ian'})
-[:KNOWS]->
(emil)
```

Ve vzorci jsme každý uzel připojili ke svému identifikátoru. Pomocí jména vlastnosti (*name*) a štítku (*Person)* (10, vlastní překlad).

2.3.7 Základní klauzule jazyku Cypher

Jako většina dotazovacích jazyků, Cypher je složen z klauzulí. V následujících kapitolách vysvětlíme ty základní. Většina klauzulí má základ převzat z jazyka SQL.

Match a Return

*„Nejjednodušší dotazy obsahují klauzuli **MATCH** následující klauzulí **RETURN**. Nejdříve si uvedeme příklad dotazu s těmito klauzulemi:*

```
MATCH (a:Person {name:'Jim'})-[:KNOWS]->(b)-[:KNOWS]-
>(c), (a)-[:KNOWS]->(c)

RETURN b, c.“ (10, str. 16, vlastní překlad)
```

Prvně je potřeba si nejdříve uvědomit o jaké uzly máme zájem. V tomto případě to jsou uzly `(a:Person {name: 'Jim'}), (b), (c), (a)`. Jak je možné vidět první uzel má

použitelnou značku (label) **Person** a vlastnost **name**. Poté pomocí ASCII nakreslíme vztahy`-[:KNOWS]->` mezi nimi (10, vlastní překlad).

V našem ukázkovém dotazu se tedy ptáme na existenci uzlu se značkou label a vlastností „Jim“. Vrácená hodnota je přiřazena ukazateli `a`, tento ukazatel nás odkazuje na daný uzel (10, vlastní překlad).

Nakonec máme klauzuli **RETURN,** která specifikuje, jaké informace (uzly a vztahy) mají být zaslány uživateli. V našem případě jde o uzly `b` a `c` (10, vlastní překlad).

Where

Poskytuje podmínky pro filtrování výsledných zobrazených dat podobně jako u SQL (8, vlastní překlad).

Create a Create unique

Vytváří uzly a vztahy grafové databáze (11, vlastní překlad).

Delete

Maže součásti grafu tedy uzly, vztahy nebo cesty. S právě mazaným uzlem musí být smazány i vztahy s ním asociovaným (11, vlastní překlad).

Merge

Vyhledává existující uzly a spojuje je, nebo vytváří nová data a ta následně spojí. V podstatě jde o kombinaci již zmíněných **MATCH** a **CREATE** (11, vlastní překlad).

Set

Je používána k upravování značení uzlů a vlastností uzlů a vztahů (11, vlastní překlad).

Foreach

Klauzule upravuje data v celém seznamu, ať už jde o součásti cesty či výsledek agregace (11, vlastní překlad).

Union

Kombinuje výsledky více různých dotazů-do jednoho výsledku. Může být využit k odstranění duplikací (11, vlastní překlad).

With

Tato klauzule umožňuje části dotazů být spojeny do jednoho, přičemž použije výsledek z první části dotazu jako kritéria pro ostatní části (11, vlastní překlad).

2.4 Shrnutí

V této kapitole jsem hierarchicky popsal jednotlivé pojmy týkající se tématu mé práce. Nejdříve jsem definoval základní pojmy jako data, databázový systém a databáze.

Následně, abych mohl analyzovat současný stav relační databáze jsem definoval všechny potřebné pojmy týkající se relačních databází jako jsou entita, atributy, tabulky apod. Nakonec jsem vypsal základní klauzule jazyka SQL, které použiji při analýze databáze.

V třetí části týkající se grafových databází jsem opět definoval nejdříve základní pojmy jako jsou vrchol, vztah a jiné. Dále jsem popsal, jak postupovat při tvorbě této databáze a jak poté dotazovat na data v ní uložená. Na konci jsem opět vypsal základní klauzule, které využiji pro vytvoření databáze.

3 Analýza současného stavu

V následující kapitole bude provedena analýza současného stavu. V případě této práce současným stavem je relační databáze, u které popíšeme jednotlivé tabulky, jejich účel a následně popíšeme nevýhody užití relační databáze.

3.1 Popis práce s databází z pohledu uživatele

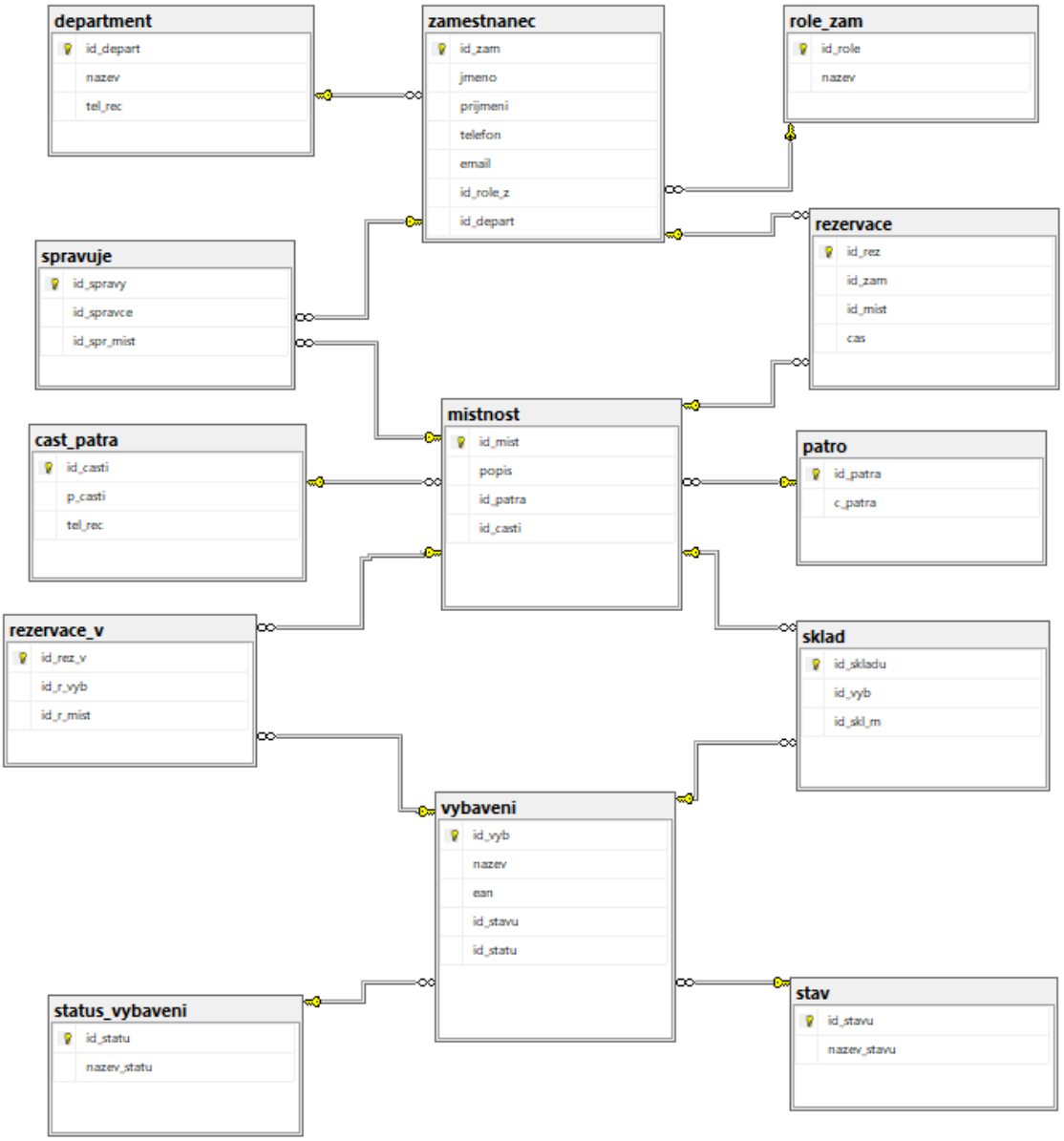
Databáze slouží k rezervaci jednotlivých místností pro účely zaměstnanců jako například pro meetingy, workshopy apod. Místnosti si uživatelé mohou vybírat i na základě vybavení jenž se v místnosti nachází. V případě rezervace místnosti se rovněž zarezervuje vybavení v této místnosti se nacházející, jako projektory a laptopy. U vybavení lze nalézt, zdali je k dispozici, je již vypůjčen nebo je v opravě. Dojde-li k nějakým potížím, databáze poskytuje informace o zaměstnanci, který danou místnost spravuje.

Když uživatel chce zarezervovat místnost přihlásí se do interního systému společnosti, zde zadá místnost a datum dne kdy chce místnost rezervovat a systém mu zobrazí volné časy možné k rezervování. Ve zvláštní sekci rezervování pak může vyčíst dostupné vybavení v místnosti. V další takové sekci pak nalezne informace o správci místnosti, kterého může v případě potřeby kontaktovat.

3.2 Informace o databázi

V této části popíši databázi z technické stránky. Databáze je vytvořena v programu Microsoft SQL Server 2016. Skládá se ze 13 tabulek (popis tabulek se nachází v následujících kapitolách: **department, zamestnanec, role_zam, spravuje, rezervace, mistnost, cast_patra, patro, rezervace_v, sklad, vybaveni, status_vybaveni** a **stav**.

Každá tabulka má jeden unikátní primární klíč a případně klíče cizí. Tabulky **spravuje**, **rezervace**, **rezervace_v** a **sklad** jsou rozkladem M:N vazeb. Všechny tabulky i s jejich atributy jsou zobrazeny v ER diagramu na obrázku č. 10.



Obrázek č. 10: ER diagram databáze (vlastní)

Přímý přístup k databázi mají jen lidé z development oddělení a správci databáze. Ostatní uživatelé přistupují k datům přes desktopové a webové aplikace jichž prostřednictvím přidávají a odebírají data. Tato data jsou však omezena těmito aplikacemi.

V databázi se nenacházejí žádné procedury, které jsou zastoupeny již zmíněnými aplikacemi, nacházejí se zde jen dva spouštěče (triggery) k nastavení statusu vybavení při vložení rezervace.

Úpravy databáze mají na starosti jen správci databáze. Ti mají za úkol jen opravovat případné chyby v databázi nebo datech, vzniklé lidským pochybením. Přímo do struktury databáze však zásadně nezasahují.

Většina aplikací využívaných pro chod databáze jsou přímo naprogramovány pro potřeby společnosti nebo jsou zakoupeny licence pro celou společnost a jsou propojeny s databází skrz desktopové prostředí. Mezi aplikace druhého zmíněného typu je například aplikace Outlook sloužící k vkládání datumů a času rezervace.

3.3 Popis jednotlivých tabulek

V této části popíšu jednotlivé tabulky a jejich vztahy na ostatní tabulky.

3.3.1 Role_zam

Udává, jakou roli zaměstnanec ve firmě zastává.

Tabulka č. 3: Tabulka role_zam (vlastní)

Atribut	Popis	Datový typ
id_role	primární klíč	integer
nazev	název role	varchar

3.3.2 Department

Obsahuje seznam departmentů

Tabulka č. 4: Tabulka department (vlastní)

Atribut	Popis	Datový typ
id_depart	primární klíč	integer
nazev	název daného departmentu	varchar
tel_rec	číslo na recepci departmentu	integer

3.3.3 Zamestnanec

Obsahuje seznam všech zaměstnanců společnosti.

Tabulka č. 5: Tabulka zamestnanec (vlastni)

Atribut	Popis	Datový typ
id_zam	primární klíč	integer
jmeno	křestní jméno	varchar
prijmeni	příjmení	varchar
telefon	služební telefon zamestnance	integer
email	služební email zaměstnance	text
id_role	cizí klíč tabulky role_zam roli zaměstnance ve společnosti	integer
id_depart	cizí klíč tab. department udává pod jaký department zaměstnanec spadá	integer

Jako primární klíč zaměstnance by se dalo použít rodné číslo, avšak to není vhodné z právních hledisek

Na tabulku zamestnanec jsou napojeny tabulky department a role_zam vazbami 1:M viz obrázek č. 11.

Obrázek č. 11: Vazby tabulky zamestnanec (vlastní)

3.3.4 Cast_patra

Udává písmeno budovy, ve které se místnost nachází, každé patro protíná dvě budovy.

Tabulka č. 6: Tabulka cast_patra (vlastní)

Atribut	Popis	Datový typ
id_casti	primární klíč	integer
nazev	označení dané budovy	varchar
tel_rec	telefon na recepci v dané budově	integer

3.3.5 Patro

Číslo patra, ve které se místnost nachází.

Tabulka č. 7: Tabulka patro (vlastní)

Atribut	Popis	Datový typ
id_patra	primární klíč	tinyint
c_patra	číslo daného patra	tinyint

3.3.6 Mistnost

Tabulka obsahuje informace o jednotlivých místnostech, může jít o kancelářské místnosti, sklady nebo místnosti pro spravující personál.

Tabulka č. 8: Tabulka mistnost (vlastní)

Atribut	Popis	Datový typ
id_mist	primární klíč	integer
popis	popis dané místnosti (např. typ, rozměry…)	text
id_patra	cizí klíč tabulky patro	tinyint
id_casti	cizí klíč tabulky části patra	integer

Tabulky patro a cast_patra jsou spojeni s tabulkou mistnost relacemi 1:M jak je zobrazeno na obrázku č. 12.

Obrázek č. 12: Vazby tabulky mistnost (vlastní)

3.3.7 Status_vybaveni

Tato tabulka obsahuje informace, jaký status dané vybavení má (např. je vypůjčen, v opravě, dostupný atd.).

Tabulka č. 9: Tabulka status_vybaveni (vlastní)

Atribut	Popis	Datový typ
id_statu	primární klíč	tinyint
nazev_statu	udává status daného vybavení	varchar

3.3.8 Stav

Tabulka stav udává, v jakém stavu vybavení je (např. nové, poškozeno atd.).

Tabulka č. 10: Tabulka stav (vlastní)

Atribut	Popis	Datový typ
id_stavu	primární klíč	tinyint
nazev_stavu	udává název stavu	varchar

3.3.9 Vybaveni

Tabulka obsahuje informace o všem dostupném vybavení.

Tabulka č. 11: Tabulka vybaveni (vlastní)

Atribut	Popis	Datový typ
id_vyb	primární klíč	integer
nazev	udává např. o jaký druh vybavení se jedná, značku a příslušný typ	varchar
ean	EAN vybavení	integer
id_stavu	cizí klíč tabulky stav	tinyint
id_statu	cizí klíč tabulky status_vybaveni	tinyint

Tabulka vybaveni je s tabulkami stav a status_vybaveni spojena relacemi 1:M jako je na obrázku č. 13.

Obrázek č. 13: Vazby tabulky vybaveni (vlastní)

3.3.10 Spravuje

Tabulka spravuje je rozkladem M:N relace mezi tabulkami zamestnanec a misnost. Slouží k spojení reálné místnosti a jejího správce.

Tabulka č. 12: Tabulka spravuje (vlastní)

Atribut	Popis	Datový typ
id_spravy	primární klíč	integer
id_spravce	cizí klíč tabulky zamestnanec udávající id správce místnosti	integer
id_spr_mist	cizí klíč tabulky mistnost udávající id spravované místnosti	integer

Je spojena s tabulkami mistnost a zamestnanec relacemi 1:M jako na obrázku č. 14.

Obrázek č. 14: Vazby tabulky spravuje (vlastní)

3.3.11 Rezervace

Tabulka rezervace je rozkladem M:N relace mezi tabulkami zamestnanec a misnost. Slouží k spojení reálné rezervované místnosti a zaměstnance, jenž ji rezervuje.

Tabulka č. 13: Tabulka rezervace (vlastní)

Atribut	Popis	Datový typ
id_rez	primární klíč	integer
id_zam	cizí klíč tabulky zamestnanec udávající id rezervujícího zaměstnance	integer
id_mist	cizí klíč tabulky mistnost udávající id rezervované místnosti	integer
zacatek	zde se zadává čas začátku rezervace	smalldatetime
konec	čas konce rezervace	smalldatetime

Je spojena s tabulkami mistnost a zamestnanec relacemi 1:M jako na obrázku č. 15.

Obrázek č. 15: Vazby tabulky rezervace (vlastní)

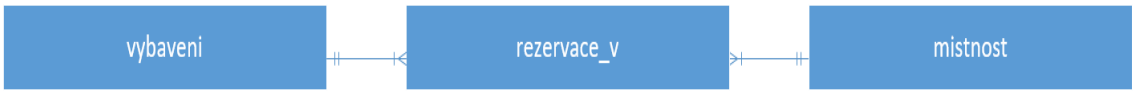
3.3.12 Rezervace_v

Slouží k rezervaci vybavení současně s rezervací.

Tabulka č. 14: Tabulka rezervace_v (vlastní)

Atribut	Popis	Datový typ
id_rez_v	primární klíč	integer
id_r_vyb	cizí klíč tabulky vybaveni	integer
id_mist	cizí klíč tabulky mistnost	integer

Je spojena s tabulkami vybaveni a mistnost relacemi 1:M jako na obrázku č. 16.



Obrázek č. 16: Vazby tabulky rezervace_v (vlastní)

3.3.13 Sklad

Obsahuje informace o tom, kde je vybavení uloženo, když není využíváno.

Tabulka č. 15: Tabulka sklad (vlastní)

Atribut	Popis	Datový typ
id_skladu	primární klíč	integer
id_vyb	cizí klíč tabulky vybaveni	integer
id_skl_m	cizí klíč tabulky mistnost	integer

Je spojena s tabulkami vybaveni a mistnost relacemi 1:M jako na obrázku č. 17.

Obrázek č. 17: Vazby tabulky sklad (vlastní)

3.4 Stav databáze

Databáze je pro chod společnosti velice důležitá proto je potřeba její správný chod. V rozrůstající se společnosti čítající tisíce zaměstnanců, desítky budov, stovky místností a přibývající množství desítek až stovek milionů dat však relační databáze začala být nedostačující v několika ohledech.

3.4.1 Rychlost načítání dat

V prvním případě jde o rychlost načítání dat. Relační databáze pro získání dat prohledávají celé tabulky (již vysvětleno v kapitole 2.3.3), ze kterých pak filtrují chtěné informace což způsobuje velké zpoždění odezvy databáze a v budoucnu by mohly nastat i větší potíže související právě s tímto zpožděním. Jedním takovým příkladem je tabulka rezervace, do které se přidávají nové a nové záznamy. Záznamy se udržují a nepromazávají pro potřeby případné kontroly stavu vybavení, místností atd. Tuto tabulku pak využívá několik aplikací a jejich dotazy na tuto tabulku mohou pak nabývat dobu i několika vteřin, což je nepřijatelná doba.

3.4.2 Komplikace u reálných vztahů

Další potíží je samotný princip relační databáze, jenž komplikuje převedení některých reálných vztahů do datové podoby. Některé reálné vztahy mezi reálnými objekty jsou se velice komplikovaně zobrazují do grafového modelu. Můžeme například uvést problém, kdy jeden ze zaměstnanců by měl více mobilních čísel. V tom případě by se musela pravděpodobně předělávat velká část databáze nebo by se tato informace nezadala do databáze což také není ideální řešení. Dalším příkladem může být jakýkoliv z M:N vztahů, kvůli kterým musí být vytvořeny zvlášť další tabulky, aby byly dekomponovány.

3.5 Shrnutí

Stav databáze je nevyhovující pro efektivní práci s daty. Nejhorším problémem je velká odezva při dotazování, kdy kvůli velkému objemu dat je čas vysoký z důvodu prohledávání veškerých dat v tabulce. Tímto problémem trpí hlavně tabulka rezervace, jenž uchovává data za několik let a při množství tisíců zaměstnanců byly v té době vytvořeny desítky milionů záznamů. Tento problém by se dal vyřešit například pohledy a indexy, avšak bylo by to jen dočasné řešení, než by se problém vrátil a mohl by se i více zhoršovat.

Dalším hůře řešitelným problémem je velmi malá flexibilita. V dnešní době, kdy vše je potřené být co nejvíce flexibilní, relační databáze v tomto ohledu zaostává. V některých případech se jen těžko zobrazují reálné vztahy do datového modelu. V lepším případě je potřeba předělat většinu datového modelu a náročně předělávat databázi což stojí čas a peníze.

4 Vlastní návrh řešení

V této části navrhnu možné řešení problematiky rezervačního systému s využitím technologie grafových databází. Cílem je vytvořit grafovou databázi, jenž by mohla nahradit již existující nedostačující databázi relační.

Nejprve se zaměřím na popsání využité technologie. Poté definuji, jak grafová databáze teoreticky řeší nedostatky popsané v analýze současného stavu. Dále popíši, jak zobrazím jednotlivé tabulky a relace v grafu grafové databáze a pokusím se sestavit konceptuální návrh databáze.

4.1 Informace o Neo4j

Neo4j je současně jedna z nejpopulárnějších grafových databází. Jedním z důvodů je, že je open source založený na jazyku java (9).

Ukládání grafové struktury je v podobě uzlů, vztahů a vlastností. Díky vlastnostem (properties) ukládaných ve vrcholech a vztazích o Neo4j mluvíme jako o *property graph database*.

Pro Neo4j jsem se rozhodl, protože jako jedna z mála databází ne-li jediná teď nové zvládne pojmout obrovské množství dat, a to až kvadriliony vrcholů a vztahů. Jedinými nevýhodami jsou samozřejmě finance. Databáze takových rozměrů jako jsou Neo4j v enterprise licencích vyjdou na statisíce dolarů ročně nehledě na potřebu koupi dostatečně výkonných serverů, jež zvládnou takové množství dat zpracovat. Můj návrh však bude nabízen firmám, které takové prostředky nepostrádají.

4.2 Řešení nedostatků relační databáze

Nejvážnějším problémem, jak už bylo dříve zmíněno je velká odezva na dotazy velkého množství dat. Tento problém u relačních databází vzniká z důvodu výpočetně náročných operací JOIN, bez kterých se relační databáze neobejde.

Grafové databáze je kategorie NoSQL databází což nám udává, že poskytují tzv. bezindexovou sousednost neboli každý uzel obsahuje přímé odkazy na své sousedy. To způsobuje nepotřebnost indexů (9).

Druhým problémem relační databáze byla malá flexibilita schématu. Grafové databáze jsou v tomto případě ideální, protože nestavějí se na fixním modelu, což zaručuje že je možné schéma neustále rozvíjet (9).

4.3 Konceptuální návrh databáze

V této části kapitoly vypracuji konceptuální návrh databáze, ve kterém jednotlivé entity, relace a případě atributy relační databáze a upravím je do podoby vhodné pro databázi grafovou.

4.3.1 Úprava jednotlivých prvků

První máme tabulku department, která kvůli optimalizaci nebude převedena do grafové databáze, bude nahrazena vlastností *Department* v uzlech s labelem **Zamestnanec**.

Stejný problém se týká tabulky **role_zam**, jenž bude též nahrazena vlastností v **Zamestnanec**. Dále potom tabulky **patro** a **cast_patra** jsou nahrazeny vlastnostmi v uzlech s labelem **Mistnost**. Stejně tak poté tabulky **status_vybaven**i a **stav** budou nahrazeny vlastnostmi v uzlech **Vybaveni**.

Nyní máme hlavní tabulky **zamestnanec**, **mistnost** a **vybaveni**. Tabulka **zamestnanec** bude uzly s označením **zamestnanec**. Ty budou deklarovány pod proměnou *„z“,* s jíž pomocí budeme uzly poté volat. Co se týče atributů tabulky zamestnanec budou zachovány všechny kromě primárního klíče a cizích klíčů tabulek role a department. Typy v grafových databázích jsou omezeny jen na text a číselné hodnoty, které se odlišují tím, že u textových typů dáme hodnoty do jednoduchých či dvojitých uvozovek (‚ nebo „“). K vlastnostem se přidávají vlastnosti *department* a *role* obě typu text.

Tabulka **mistnost** je řešena stejně jako zamestnanec tedy zůstávají všechny atributy až na primární klíč a cizí klíče tabulek patro a cast_patra. Cizí klíče jsou nahrazeny vlastnostmi patro číselného *datového typu* a *cast_patra* datového typu text. Uzly jsou nazvány stejně tedy Mistnost a jsou uloženy pod deklarovanou proměnou *„m“*.

Nakonec je tabulka **vybaveni.** Ta je řešena stejně jako tabulky mistnost a zamestnanec. Primární klíč a cizí klíče jsou odstraněny a cizí klíče jsou nahrazeny vlastnostmi *stav* a *status* obě textového datového typu. Uzly jsou nazvány vybaveni a jsou uloženy pod proměnnou *„m“*.

Nakonec máme tabulky **rezervace, spravuje, sklad** a **rezervace_v.** Tyto tabulky existují z důvodu dekompozice relací M:N, jenž nejsou v relačních databázích podporovány. Grafové databáze toto omezení nemají. Proto jsou tyto tabulky nahrazeny vztahy, které budou nést jejich názvy.

Tabulka **rezervace** bude nahrazena vztahem s labelem **rezervoval**. Tabulka má jako jediná jiné atributy kromě primárních a cizích klíčů. To však též není problém z důvodu možnosti mít ve vztazích vlastnosti. Tedy klíčové atributy budou odstraněny a do vztahu se převedou vlastnosti *zacatek* a *konec* časového datového typu. Časový datový typ je v grafových databázích nejlépe řešen pomocí textového typu. Čas se vkládá pomocí aplikací nad databází a je předáván v jednotném formátu „RRRR-MM-DD HH:MM:SS“ ten je uložen do databáze jako text. Při dotazování databáze na jakýkoliv čas je tento text pak v aplikaci převeden zpět do časového typu.

Dále máme tabulky **spravuje, sklad** a **rezervace_v.** Ty neobsahují kromě klíčů žádné jiné atributy. Proto z nich vytvořím vztahy bez vlastností. Z tabulky spravuje vztah **spravuje**, z tabulky sklad vznikne vztah **nachazi_se** a z rezervace_v vznikne **rezervoval_vyb.** Všechny vztahy pak mají svoji proměnnou. Rezervuje má *„r“*, spravuje má *„s“*, nachazi_se má *„n“* a rezervace_v má *„rv“*.

Tabulka č. 16 obsahuje výsledný přehled entit a jejich podoby v grafové databázi.

Tabulka č. 16: Přehled entit (vlastní)

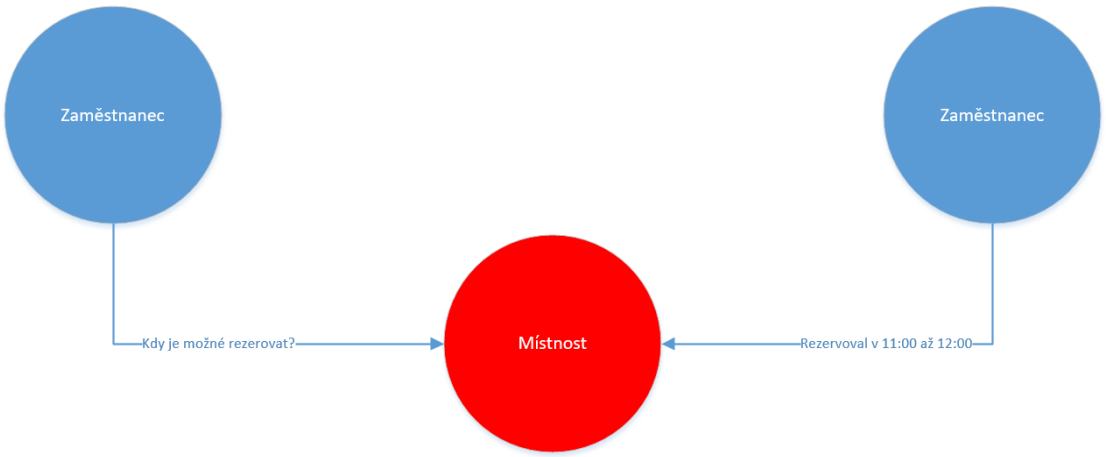
Tabulka relační databáze	Entita grafové databáze	Proměnná
zamestnanec	uzel zamestnanec	z
mistnost	uzel mistnost	m
vybaveni	uzel vybaveni	v
role_zam	vlastnost role v uzlu zamestnanec	
department	vlastnost department v uzlu zamestnanec	
patro	vlastnost patro v uzlu mistnost	
cast_patra	vlastnost cast_patra v uzlu mistnost	
status_vybaveni	vlastnost status v uzlu vybaveni	
stav	vlastnost stav v uzlu vybaveni	
rezervace	vztah rezervoval	r
spravuje	vztah spravuje	s
sklad	vztah nachazi_se	n
rezervace_v	vztah rezervoval_vyb	rv

4.3.2 Určení vztahů

Jak již bylo zmíněno v druhé kapitole využitá technologie Neo4j nepodporuje synchronní vztahy, což musíme celou dobu při vytváření databáze brát v potaz. V databázi Neo4j je se však můžeme dotazovat i na uzly proti směru vrahu což ulehčuje určení grafu.

Nejdříve máme vztah rezervoval. Nyní se musíme zeptat co uživatel bude chtít vědět při rezervaci: Jako uživatel chci vědět, jaký jiný uživatel si zarezervoval tuto místnost, abych mohl najít volný čas pro moji rezervaci.

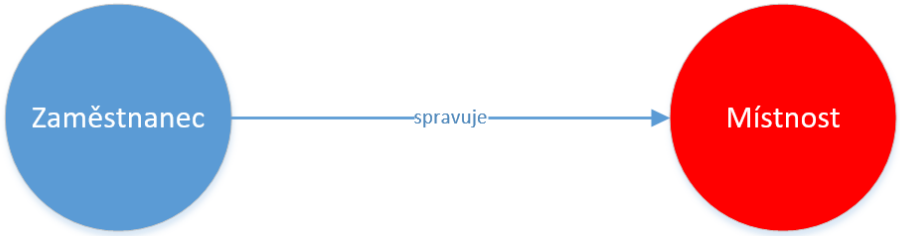
Tedy vyhledám ostatní uživatele jako na obrázku abych se podíval do jejich vztahů *rezervoval* a zjistil, který čas je volný. To nám ukazuje, jakým směrem půjde vztah rezervoval. Viz. obrázek č.18.



Obrázek č. 18: Vztah rezervoval (vlastní)

S ostatními vztahy uděláme to samé. V tomto případě se nám nabízí možnost *zaměstnanec spravuje* nebo *místnost je spravována*. Aby nevznikl synchronní vztahy se vztahem rezervoval, rovnou můžeme určit, že vytvoříme druhý typ vztahu stejným směrem.

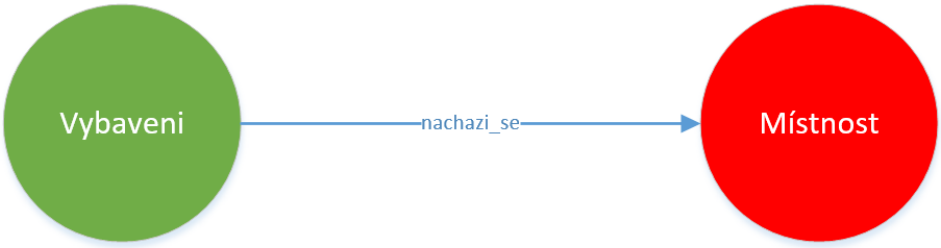
Z toho nám vychází vztah na obrázku č. 19.



Obrázek č. 19: Vztah spravuje (vlastní)

Pro další vztah **nachazi_se** si položíme otázku: Jako uživatel chci vědět, kde se vybavení, které potřebuji pro svou činnost, právě nachází, abych mohl místnost rezervovat.

Tím nám vznikne vztah na obrázku č. 20.



Obrázek č. 20: Vztah nachazi_se (vlastní)

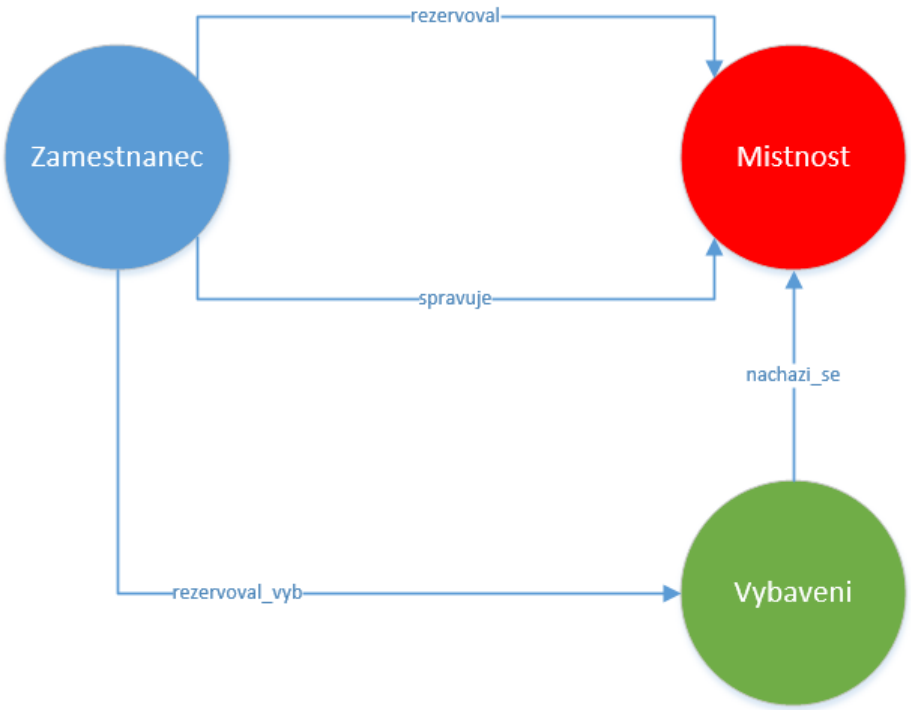
Nakonec máme vztah **rezervoval_vyb**, na který je stejná otázka jako na vztah *rezervoval* jen mezi uzly *zamestnanec a vybaveni*. Tento vztah vznikne automaticky aplikací při zarezervování místnosti, ve které se vybavení nachází.

Vztah **rezervoval_vyb** se nachází na obrázku č. 21.

Obrázek č. 21: Vztah rezervoval_vyb (vlastní)

4.3.3 Výsledný graf databáze

Na obrázku č. 22 můžeme vidět výsledný graf databáze jenž můžeme použít pro náš návrh.



Obrázek č. 22: Výsledný graf (vlastní)

4.4 Tvorba dotazů v Cypher

Teď když jsem stanovil graf databáze mohu začít s tvořením dotazů pro tvorbu jednotlivých entit databáze. Prvně je potřeba vytvořit vrcholy a až poté jsou k nim tvořeny vztahy.

4.4.1 Tvorba vrcholů

Ukáži tvorbu vrcholů na příkladu tvorby jednoho vrcholu s label *zamestnanec*. Pro vytvoření vrcholů potřebujeme jen jednu klauzuli, a to klauzuli CREATE. Dotaz se tvoří zadáním CREATE dále použijeme závorky, poté deklarujeme proměnnou. V našem případě použiji proměnnou *„z“*. Tato proměnná se deklaruje jen u prvního uzlu daného typu. U ostatních uzlů s label zamestnanec to již není potřeba. Prozatím jsem napsal tento dotaz:

```
CREATE (z: zamestnanec)
```

Dále použiji složené závorky, které slouží k vytvoření vlastností uzlů. Následuje již jen výpis jednotlivých názvů vlastností a jejich hodnoty. Textové datové typy jsou vloženy do uvozovek, zatímco číselné jsou bez uvozovek. Výsledný dotaz bude vypadat takto:

```
CREATE
      (
      z: zamestnanec
                  {
                  jmeno: 'Karl',
                  prijmeni: 'Berghof',
                  telefon: 126999989,
                  email: 'rtyityud@dasd.cz',
                  role: 'CEO',
                  department: 'CEO'
                  }
      )
```

Vzorec vytvoří uzel, který bude zastupovat zaměstnance se jménem Karl Berghof.

4.4.2 Tvorba vztahů

Tvoření vztahů je možné mnoha způsoby, které se používají v různých situacích. V mojí situaci, kdy jsem jako první vytvořil uzly teď mohu použít způsob ve kterém pomocí klauzule MATCH vyberu uzly mezi jež chci vytvořit vztah a ten pomocí klauzule CREATE vytvořím. Uvedu si to na příkladu, kdy chci vytvořit vztah *rezervoval* mezi uzlem *zamestnanec* a uzlem *misnost*.

Nejprve tedy napíši klauzuli MATCH, po které napíši proměnné a labely uzlů na které se chci dotázat. Ty napíši stejně jako při vytváření uzlu:

```
MATCH (z:zamestnanec). (m:mistnost)
```

Dále musím uvést podmínky podle, kterých chci najít určité uzly. K tomu využiji klauzuli WHERE. Jelikož budu hledat více uzlů, což znamená více podmínek při vyhledávání, použiji i logický operátor AND mezi podmínkami. Při rezervaci se bude muset zaměstnanec přihlašovat do systému a zadá místnost jež chce rezervovat. Je tedy možné použít jeho křestní jméno a příjmení pro vyhledání zaměstnance v databázi a patro a část patra pro nalezení místnosti:

```
WHERE z.prijmeni = 'Berghof' AND z.jmeno:'Karl' AND m.patro = 1 AND
m.cast_patra = 'B'
```

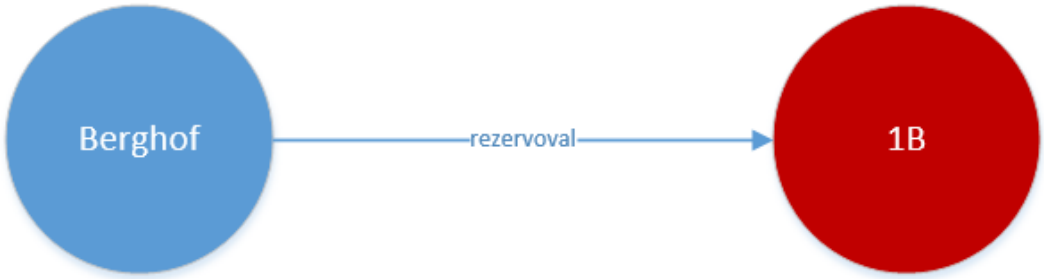
Nakonec když jsem vyhledal uzly, mezi něž chci vytvořit vztah, nyní můžu přistoupit k jeho tvorbě. Uzly již zastoupí jen proměnné *z* a *m*, dále pomocí ASCII znaků popíši směr vztahu. Vztah se tvoří pomocí hranatých závorek, do nichž je opět deklarována proměnná tedy v našem případě proměnná *„r“*. Dále už jen zbývá doplnit vlastnosti do vztahu opět pomocí složených závorek. U vztahů s label *rezervoval* jsou to vlastnosti udávající začátek a konec rezervace.

Nakonec pomocí klauzule RETURN a proměnné *r* si nechám vypsat vytvořený vztah a jeho vlastnosti.

Konečný dotaz bude vypadat takto:

```
MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Berghof' AND z.jmeno:'Karl' AND m.patro = 1 AND
m.cast_patra = 'B'
CREATE (z)-[r:rezervoval {zacatek: '2018-03-25 11:00:00', konec:
'2018-03-25 12:00:00'}]->(m)
RETURN r
```

Na obrázku č. 23 je zobrazen mnou vytvořený vztah mezi uzly.



Obrázek č. 23: Vytvořený vztah (vlastní)

4.4.3 Rozšíření databáze o featury

Součástí původní databáze byly spouště, jež měly za úkol automaticky měnit statusy vybavení. V případě grafových databází spouště, procedury a podobné featury jsou doporučeny řešit v aplikacích nad databází. Přímo v databázi by bylo tedy možné jen připravit dotazy využitelné pro vložení do kódu programu. Proto se jimi tato práce nezabývá.

4.5 Zhodnocení stavu databáze

Cílem této práce bylo zanalyzovat relační databázi již fungující ve společnosti, definovat nedostatky této databáze a následně přijít s řešením nové databáze, která tyto nedostatky pokryje.

Prvním problémem byla rychlost odezvy databáze. Ta byla vyřešena zvolením jiné databázové technologie tedy grafové databáze. Tato problematika byla právě jedním z důvodů proč tato technologie byla vyvinuta. Grafová databáze v porovnání s relační má i s přibývajícím množstvím dat odezvu konstantní.

Dalším problémem byla nedostatečná flexibilita struktury databáze. Grafová databáze je tzv. bezstrukturová. Nemá žádnou fixní strukturu a je možné ji dále bez jakýchkoliv obtíží upravovat a rozšiřovat což u relačních databází je v některých případech komplikované.

Posledním problémem bylo obtížné převedení některých reálných vztahů do databáze. Uvedené příklady jsou vyřešeny již v samotném principu grafové databáze. V případě více telefonních čísel u jednoho zaměstnance by se jen vytvořila další vlastnost což by nijak neovlivnilo ostatní uzly. Nakonec problematika M:N vztahů je též již vyřešena jelikož graf tyto vztahy podporuje a je možné je vytvořit.

ZÁVĚR

Cílem této práce bylo zanalyzovat již fungující relační databázi, zjistit její nedostatky a následně navrhnout jejich řešení.

V první části práce jsem vypracoval potřebná teoretická východiska potřebná pro následující části práce.

Druhá část obsahovala analýzu současného stavu, ve které jsem nejdříve uvedl informace, o již existující databázi, jak s ní běžný zaměstnanec pracuje, a nakonec vypsal a posoudil výhody a nedostatky této databáze.

V poslední kapitole jsem se zabýval vlastním návrhem databáze. Nejdříve jsem popsal technologii grafické databáze. Následně jsem všechny entity databáze původní převedl do entit grafu. Nakonec jsem zhodnotil, zda a nakolik nová technologie vyřešila nedostatky technologie původní.

Nová databáze zásadně urychlí práci s daty, zmenší náročnost operací a nabídne možnosti upravení nebo rozšíření databáze v budoucnu.

Podle výše uvedených informací tato práce splnila své cíle.

SEZNAM POUŽITÝCH ZDROJŮ

(1) OPPEL, Andrew a David KRÁSENSKÝ. *Databáze bez předchozích znalostí*. Vyd. 1. Brno: Computer Press, 2006. ISBN 80-251-1199-7.

(2) ŠIMŮNEK, Milan. *SQL: kompletní kapesní průvodce*. Vyd. 1. Praha: Grada, 1999. ISBN 80-7169-692-7.

(3) LUHAN, Jan a Jiří KŘÍŽ. *Databázové systémy: Databázové systémy a architektury DBMS, metodologie návrhu DB* [přednáška]. Brno, 30. 9. 2015.

(4) OPPEL, Andrew J. *SQL bez předchozích znalostí: [průvodce pro samouky]*. Brno: Computer Press, 2008. ISBN 978-80-251-1707-1.

(5) HARRINGTON, Jan L. *Relational database design and implementation: clearly explained*. 4th edition. Cambridge, MA: Elsevier, 2016. ISBN 9780128043998.

(6) KŘÍŽ, Jiří. *Dotazovací jazyk SQL: Databázové systémy* [přednáška]. Brno, 15. 10. 2015.

(7) GARCIA-MOLINA, H., J. D. ULLMAN and J. WIDOM. *Database systems: The Complete Book*. 2nd ed. Upper Saddle River, N.J.: Pearson Prentice Hall, 2008. 1248 p. ISBN 978-0131873254.

(8) ROBINSON, I., J. WEBBER a E. EIFREM. *Graph databases: New Opportunities for Connected Data*. 2nd ed. Beijing: O'Reilly Media, 2015. 238 p. ISBN 978-1491930892.

(9) RAMBA, Jaroslav. Modelování dat v Neo4j: 20.11. 2016. *Jaroslav Ramba-Data Analyst and Developer* [online]. 20.11.2016 [cit. 2018-03-21]. Dostupné z: https://www.ramba.cz/blog/navrh-modelu-grafove-databaze-neo4j#

(10) SASAKI, Bryce Merkl, Joy CHAO a Rachel HOWARD. *Graph Databases for Beginners* [pdf]. Neo4j, 2018 [cit. 2018-03-21]. Dostupné z: https://neo4j.com/

(11) *The Neo4j Developer Manual v3.3* [online]. Neo4j, 2017 [cit. 2018-04-07]. Dostupné z: https://neo4j.com/docs/developer-manual/3.3/

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

Příloha 1: SQL Query

```
create database test
go

use test
go

create schema bp authorization dbo
go

create table department (
       id_depart int NOT NULL IDENTITY(1,1) PRIMARY KEY,
       nazev varchar(50) NOT NULL, tel_rec int)
create table role_zam (
       id_role int NOT NULL IDENTITY(1,1) PRIMARY KEY,
       nazev varchar(50) NOT NULL)
create table patro (
       id_patra tinyint NOT NULL IDENTITY(1,1) PRIMARY KEY,
        c_patra tinyint NOT NULL)
create table cast_patra (
       id_casti int NOT NULL IDENTITY(1,1) PRIMARY KEY,
       p_casti varchar(2) NOT NULL, tel_rec int)
create table stav (
       id_stavu tinyint NOT NULL IDENTITY(1,1) PRIMARY KEY,
       nazev_stavu varchar(20) NOT NULL)
create table status_vybaveni (
       id_statu tinyint NOT NULL IDENTITY(1,1) PRIMARY KEY,
       nazev_statu varchar(20) NOT NULL)
create table zamestnanec (
       id_zam int NOT NULL IDENTITY(1,1) PRIMARY KEY,
       jmeno varchar(20) NOT NULL, prijmeni varchar(30) not null,
       telefon int,
       email text,
       id_role_z int not null,
       id_depart int not null,
       CONSTRAINT FK_role FOREIGN KEY (id_role_z) REFERENCES role_zam(id_role),
       CONSTRAINT FK_Depart FOREIGN KEY (id_depart) REFERENCES
       department(id_depart))
create table vybaveni (
       id_vyb int NOT NULL IDENTITY(1,1) PRIMARY KEY,
       nazev varchar(50) NOT NULL,
       ean int,
       id_stavu tinyint not null,
       id_statu tinyint not null,
       CONSTRAINT FK_status FOREIGN KEY (id_statu) REFERENCES
       status_vybaveni(id_statu),
       CONSTRAINT FK_stav FOREIGN KEY (id_stavu) REFERENCES stav(id_stavu))
create table mistnost (
       id_mist int NOT NULL IDENTITY(1,1) PRIMARY KEY,
       popis text,
       id_patra tinyint not null,
       id_casti int not null,
       CONSTRAINT FK_patro FOREIGN KEY (id_patra) REFERENCES patro(id_patra),
       CONSTRAINT FK_cast FOREIGN KEY (id_casti) REFERENCES cast_patra(id_casti))
```

```
create table rezervace (
       id_rez int not null IDENTITY(1,1) PRIMARY KEY,
       id_zam int not null,
       id_mist int not null,
       zacatek smalldatetime not null,
       konec smalldatetime not null,
       CONSTRAINT FK_id_zam FOREIGN KEY (id_zam) REFERENCES zamestnanec(id_zam),
       CONSTRAINT FK_mist FOREIGN KEY (id_mist) REFERENCES mistnost(id_mist))

create table spravuje (
       id_spravy int not null IDENTITY(1,1) PRIMARY KEY,
       id_spravce int not null,
       id_spr_mist int not null,
       CONSTRAINT FK_id_spravce FOREIGN KEY (id_spravce) REFERENCES
       zamestnanec(id_zam),
       CONSTRAINT FK_spr_mist FOREIGN KEY (id_spr_mist) REFERENCES
       mistnost(id_mist))

create table sklad (
       id_skladu int not null IDENTITY(1,1) PRIMARY KEY,
       id_vyb int not null,
       id_skl_m int not null,
       CONSTRAINT FK_id_vyb FOREIGN KEY (id_vyb) REFERENCES vybaveni(id_vyb),
       CONSTRAINT FK_id_skl FOREIGN KEY (id_skl_m) REFERENCES mistnost(id_mist))

create table rezervace_v (
       id_rez_v int not null IDENTITY(1,1) PRIMARY KEY,
       id_r_vyb int not null,
       id_r_mist int not null,
       CONSTRAINT FK_r_vyb FOREIGN KEY (id_r_vyb) REFERENCES vybaveni(id_vyb),
       CONSTRAINT FK_r_mist FOREIGN KEY (id_r_mist) REFERENCES mistnost(id_mist))

insert into dbo.department (nazev,tel_rec) values ('CEO',456745523)
insert into dbo.department (nazev,tel_rec) values ('chiefs',456745523)
insert into dbo.department (nazev,tel_rec) values ('development',456666813)
insert into dbo.department (nazev,tel_rec) values ('testing',456789813)
insert into dbo.department (nazev,tel_rec) values ('linguistic',455456813)
insert into dbo.department (nazev,tel_rec) values ('human resources',455459813)
insert into dbo.department (nazev,tel_rec) values ('administration',456745523)

insert into dbo.role_zam (nazev) values ('CEO')
insert into dbo.role_zam (nazev) values ('chief')
insert into dbo.role_zam (nazev) values ('project manager')
insert into dbo.role_zam (nazev) values ('test leader')
insert into dbo.role_zam (nazev) values ('development leader')
insert into dbo.role_zam (nazev) values ('linguistic leader')
insert into dbo.role_zam (nazev) values ('HR manager')
insert into dbo.role_zam (nazev) values ('senior')
insert into dbo.role_zam (nazev) values ('junior')
insert into dbo.role_zam (nazev) values ('temporary')
insert into dbo.role_zam (nazev) values ('service stuff')
insert into dbo.role_zam (nazev) values ('secretary')

insert into dbo.patro(c_patra) values (0)
insert into dbo.patro(c_patra) values (1)
insert into dbo.patro(c_patra) values (2)
insert into dbo.patro(c_patra) values (3)
insert into dbo.patro(c_patra) values (4)
insert into dbo.patro(c_patra) values (5)
```

```
insert into dbo.cast_patra(p_casti) values ('A')
insert into dbo.cast_patra(p_casti) values ('B')
insert into dbo.cast_patra(p_casti) values ('C')
insert into dbo.cast_patra(p_casti) values ('D')

insert into dbo.stav(nazev_stavu) values ('new')
insert into dbo.stav(nazev_stavu) values ('used')
insert into dbo.stav(nazev_stavu) values ('being repaired')
insert into dbo.stav(nazev_stavu) values ('broken')

insert into dbo.status_vybaveni(nazev_statu) values ('available')
insert into dbo.status_vybaveni(nazev_statu) values ('reservated')
insert into dbo.status_vybaveni(nazev_statu) values ('borrowed')
insert into dbo.status_vybaveni(nazev_statu) values ('unavailable')

insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Karl', 'Berghof',126999989,'rtyityud@dasd.cz',1,6)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Julia', 'Thart',126777789,'tyuitysd@dasd.cz',2,7)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Mark', 'Austahg',126666789,'iyurrtysd@dasd.cz',3,1)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Zac', 'Adcsd',145356789,'rtuyryasd@dasd.cz',4,2)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Emily', 'Asdfg',126456789,'asuyiyusdsd@dasd.cz',5,3)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Monica', 'Qwert',126456789,'asiyrrtd@dasd.cz',6,4)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Jessica', 'Zxcv',124546489,'ityityiasd@dasd.cz',7,5)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Mildred', 'Fghj',12963449,'kuykyjsd@dasd.cz',11,7)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Ben', 'Tyui',126556789,'atytyjd@dasd.cz',11,7)
insert into dbo.zamestnanec(jmeno, prijmeni, telefon,email, id_role_z, id_depart)
values ('Thomas', 'Cvbm',126444789,'werwesd@dasd.cz',11,7)

insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Projektor 456',
4576782,1,1)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Projektor 456',
4527682,1,1)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Projektor 456',
44852,1,1)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Projektor 456',
45468782,1,1)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Projektor 456',
457545282,1,2)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Notebook 789',
44458782,1,2)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Notebook 789',
452428782,1,1)
insert into dbo.vybaveni(nazev, ean, id_stavu,id_statu) values ('Notebook 789',
456453782,1,1)

insert into dbo.mistnost(popis, id_patra,id_casti) values ('Meeting room',1,1)
insert into dbo.mistnost(popis, id_patra,id_casti) values ('Meeting room',2,2)
insert into dbo.mistnost(popis, id_patra,id_casti) values ('Meeting room',3,3)
insert into dbo.mistnost(popis, id_patra,id_casti) values ('Meeting room',4,4)
insert into dbo.mistnost(popis, id_patra,id_casti) values ('Meeting room',5,2)
```

```
insert into dbo.rezervace(id_zam,id_mist,zacatek,konec) values (5,1,'2018-12-13
11:00:00','2018-12-13 12:00:00')
insert into dbo.rezervace(id_zam,id_mist,zacatek,konec) values (4,2,'2018-12-13
10:00:00','2018-12-13 11:00:00')
insert into dbo.rezervace(id_zam,id_mist,zacatek,konec) values (3,3,'2018-12-14
15:00:00','2018-12-14 16:00:00')

insert into dbo.spravuje(id_spravce,id_spr_mist) values (8,1)
insert into dbo.spravuje(id_spravce,id_spr_mist) values (9,2)
insert into dbo.spravuje(id_spravce,id_spr_mist) values (10,3)

insert into dbo.sklad(id_vyb,id_skl_m) values (1,4)
insert into dbo.sklad(id_vyb,id_skl_m) values (2,5)
insert into dbo.sklad(id_vyb,id_skl_m) values (3,1)
insert into dbo.sklad(id_vyb,id_skl_m) values (4,2)

insert into dbo.rezervace_v(id_r_vyb,id_r_mist) values (3,1)
insert into dbo.rezervace_v(id_r_vyb,id_r_mist) values (4,2)
go

-----TRIGGERS--------------------------------------------------

create trigger dbo.zmena_statu_reservated on dbo.rezervace_v
for insert
as begin
            declare      @id_vybaveni int
            select @id_vybaveni = id_r_vyb from inserted
            update dbo.vybaveni set id_statu = 2 where id_vyb = @id_vybaveni
            select * from inserted where @id_vybaveni = id_r_vyb
end
go

create trigger dbo.zmena_statu_available on dbo.rezervace_v
for delete
as begin
            declare      @id_vybaveni int
            select @id_vybaveni = id_r_vyb from deleted
            update dbo.vybaveni set id_statu = 1 where id_vyb = @id_vybaveni
            select * from deleted where @id_vybaveni = id_r_vyb
end
go
```

Příloha 2: Cypher query

```
-------UZLY
create (z:zamestnanec {jmeno:'Karl',  prijmeni: 'Berghof',
telefon:126999989,email: 'rtyityud@dasd.cz',role: 'CEO', department:'CEO'})
create (:zamestnanec {jmeno:'Julia',  prijmeni: 'Thart', telefon:126777789,email:
'tyuitysd@dasd.cz',role: 'CEO', department:'CEO'})
create (:zamestnanec {jmeno:'Mark',  prijmeni: 'Austahg',
telefon:126666789,email: 'iyurrtysd@dasd.cz',role: 'chiefs', department:'CEO'})
create (:zamestnanec {jmeno:'Zac',  prijmeni: 'Adcsd', telefon:145356789,email:
'rtuyryasd@dasd.cz',role: 'Human resources administrator', department:'HR'})
create (:zamestnanec {jmeno:'Emily',  prijmeni: 'Asdfg', telefon:126456789,email:
'asuyiyusdsd@dasd.cz',role: 'Linguist', department:'Linguistic'})
create (:zamestnanec {jmeno:'Monica',  prijmeni: 'Qwert',
telefon:126456789,email: 'asiyrrtd@dasd.cz',role: 'Junior Tester',
department:'Testing'})
create (:zamestnanec {jmeno:'Monica',  prijmeni: 'Zxcv', telefon:124546489,email:
'ityityiasd@dasd.cz',role: 'Senior Developer', department:'Developers'})
create (:zamestnanec {jmeno:'Monica',  prijmeni: 'Fghj', telefon:12963449,email:
'kuykyjsd@dasd.cz',role: 'Administrator', department:'Administrative'})
create (:zamestnanec {jmeno:'Monica',  prijmeni: 'Tyui', telefon:126556789,email:
'atytyjd@dasd.cz',role: 'Administrator', department:'Administrativ'})
create (:zamestnanec {jmeno:'Monica',  prijmeni: 'Cvbm', telefon:126444789,email:
'werwesd@dasd.cz',role: 'Administrator', department:'Administrativ'})

create (v:vybaveni {nazev:'Projektor 456', ean: 4576782, stav: 'New',status:
'Available'})
create (:vybaveni {nazev:'Projektor 456', ean: 44852, stav: 'New',status:
'Available'})
create (:vybaveni {nazev:'Projektor 456', ean: 45468782, stav: 'New',status:
'Available'})
create (:vybaveni {nazev:'Projektor 456', ean: 457545282, stav: 'New',status:
'Reservated'})
create (:vybaveni {nazev:'Notebook 789', ean: 44458782, stav: 'New',status:
'Reservated'})

create (m:mistnost {popis: 'Meeting room', patro: 0,cast_patra: 'A'})
create (:mistnost {popis: 'Meeting room', patro: 1,cast_patra: 'B'})
create (:mistnost {popis: 'Meeting room', patro: 2,cast_patra: 'C'})
create (:mistnost {popis: 'Meeting room', patro: 3,cast_patra: 'D'})
create (:mistnost {popis: 'Meeting room', patro: 4,cast_patra: 'B'})

-------VZTAHY
-------REZERVACE

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Berghof' AND m.patro = 1 and m.cast_patra = 'B'
CREATE (z)-[r:rezervoval { zacatek: '2018-03-25 11:00:00', konec: '2018-03-25
12:00:00' }]->(m)
RETURN r

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Qwert' AND m.patro = 2 and m.cast_patra = 'C'
CREATE (z)-[r:rezervoval { zacatek: '2018-05-22 12:00:00', konec: '2018-05-22
15:00:00' }]->(m)
RETURN r

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Tyui' AND m.patro = 2 and m.cast_patra = 'C'
```

```
CREATE (z)-[r:rezervoval { zacatek: '2018-05-10 11:00:00', konec: '2018-05-10
12:00:00' }]->(m)
RETURN r

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Tyui' AND m.patro = 3 and m.cast_patra = 'D'
CREATE (z)-[r:rezervoval { zacatek: '2018-05-10 11:00:00', konec: '2018-05-10
12:00:00' }]->(m)
RETURN r

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Cvbm' AND m.patro = 3 and m.cast_patra = 'D'
CREATE (z)-[r:rezervoval { zacatek: '2018-05-10 14:00:00', konec: '2018-05-10
15:00:00' }]->(m)
RETURN r

--------SPRAVUJE

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Fghj' AND m.patro = 3 and m.cast_patra = 'D'
CREATE (z)-[s:spravuje]->(m)
RETURN s


MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Tyui' AND m.patro = 2 and m.cast_patra = 'C'
CREATE (z)-[s:spravuje]->(m)
RETURN s

MATCH (z:zamestnanec),(m:mistnost)
WHERE z.prijmeni = 'Cvbm' AND m.patro = 1 and m.cast_patra = 'B'
CREATE (z)-[s:spravuje]->(m)
RETURN s

----------NACHAZI_SE

MATCH (v:vybaveni),(m:mistnost)
WHERE v.ean = 4576782 AND m.patro = 1 and m.cast_patra = 'B'
CREATE (v)-[n:nachazi_se]->(m)
RETURN n

MATCH (v:vybaveni),(m:mistnost)
WHERE v.ean = 44852 AND m.patro = 2 and m.cast_patra = 'C'
CREATE (v)-[n:nachazi_se]->(m)
RETURN n

MATCH (v:vybaveni),(m:mistnost)
WHERE v.ean = 45468782 AND m.patro = 3 and m.cast_patra = 'D'
CREATE (v)-[n:nachazi_se]->(m)
RETURN n

--------REZERVACE_VYB

MATCH (z:zamestnanec {prijmeni:'Berghof'})-[r]->(m:mistnost{patro: 1
,cast_patra:'B'})<-[n]-(v:vybaveni)
CREATE (z)-[rv:rezervoval_vyb]->(v)
RETURN v
```

```
MATCH (z:zamestnanec {prijmeni:'Qwert'})-[r]->(m:mistnost{patro: 2,cast_patra:
'C'})<-[n]-(v:vybaveni)
CREATE (z)-[rv:rezervoval_vyb]->(v)
RETURN v

MATCH (z:zamestnanec {prijmeni:'Tyui'})-[r]->(m:mistnost{patro: 2,cast_patra:
'D'})<-[n]-(v:vybaveni)
CREATE (z)-[rv:rezervoval_vyb]->(v)
RETURN v

MATCH (z:zamestnanec {prijmeni:'Tyui'})-[r]->(m:mistnost{patro: 3,cast_patra:
'D'})<-[n]-(v:vybaveni)
CREATE (z)-[rv:rezervoval_vyb]->(v)
RETURN v

MATCH (z:zamestnanec {prijmeni:'Cvbm'})-[r]->(m:mistnost{patro: 3,cast_patra:
'D'})<-[n]-(v:vybaveni)
CREATE (z)-[rv:rezervoval_vyb]->(v)
RETURN v
```