Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Objektové a relační databáze

Bc. Jan Císař

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jan Císař

Systémové inženýrství a informatika
Informatika

Název práce

Objektové a relační databáze

Název anglicky

Object and relation databases

Cíle práce

Cílem práce je pomocí praktického příkladu porovnat relační a objektovou (grafovou) databázovou technologii z pohledu praktické použitelnosti. Práce ma také za úkol seznámit s moderními trendy v databázové technologii a prozkoumat rozdíly mezi nimi.

Metodika

V první části práce bude teoretický přehled použitých zdrojů ve formě literární rešerše. Druhá část práce bude praktická ve formě projektové dokumentace databázové aplikace v oblasti sportu. Konkrétně se jedná o projektovou dokumentaci v hokejovém týmu. Na první společnou analytickou část podpořenou modely ve standardu UML budou navazovat dvě různé implementace za účelem praktického porovnání konkrétní relační a konkrétní objektové (alias grafové) databázové technologie. Tato řešení budou porovnána podle alespoň tří parametrů kvality ISO/IEC 25000.

Doporučený rozsah práce

80 – 120 stran

Klíčová slova

objektová databáze, relační databáze, návrh databáze, class diagram, ER diagram

---

Doporučené zdroje informací

ČADA, Ondřej. Objektové programování: naučte se pravidla objektového myšlení. Praha: Grada, 2009. Průvodce (Grada). ISBN 978-80-247-2745-5.

HERNANDEZ, Michael J. a John VIESCAS. Myslíme v jazyku SQL: tvorba dotazů. Praha: Grada, 2004. Knihovna programátora (Grada). ISBN 80-247-0899-x.

KALUŽA, Jindřich a Ludmila KALUŽOVÁ. Modelování dat v informačních systémech. Praha: Ekopress, 2012. ISBN 978-80-86929-81-1.

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 30. 03. 2022

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Objektové a relační databáze" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.3.2022 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Vojtěch Merunka, Ph.D. za odborné rady, výstižné informace, poskytnutý čas a poznámky při zpracování diplomové práce. Dále bych rád poděkoval rodině za morální a finanční podporu po celou dobu studia. Poslední poděkování se vztahuje ke korespondentům dotazníkového šetření, kteří si našli čas k vyplnění dotazníku pro diplomovou práci.

Objektová a relační databáze

Abstrakt

Diplomová práce se zabývá tématem objektové a relační databázové technologie. Účelem práce je jejich porovnání z pohledu praktické použitelnosti. První část teoretické části se zabývá problematikou modelů ve standartu UML. Podrobněji popsané jsou modely Entity Relationship Diagram a CLASS diagram. Druhá část popisuje základy objektové a relační databáze, jednotlivé principy a porovnání. V praktické části je na základě provedené analýzy navržen konkrétní CLASS Diagram. Na jeho základu jsou vypracovány implementace pro objektovou a relační databázi týkající se praktického použití pro hokejový klub. Relační databáze je podpořena Entity-Relationship diagramem k upřesnění vazebních tabulek. Následuje dotazníkové šetření vycházející z normy ČSN/IEC 25000 na praktické porovnání vypracovaných implementací. Na základě dotazníkového šetření jsou analyzovány výsledky jednotlivých řešení.

Klíčová slova: Objektová databáze, Relační databáze, Návrh databáze, CLASS Diagram, Entity-Relationship Diagram

Object and relation databases

Abstract

This thesis deals with the topic of object and relational database technology. The purpose of this work is to compare them in terms of practical applicability. The first part of the theoretical part deals with the issue of models in the UML standard. The Entity Relationship Diagram and CLASS diagram models are described in more detail. The second part describes the basics of object and relational databases, individual principles, and comparisons. In the practical part, a specific CLASS Diagram is designed based on the performed analysis. Based on it, implementations for the object and relational database concerning practical use for the hockey club are developed. The relational database is supported by an Entity-Relationship diagram to refine the binding tables. The following is a questionnaire survey based on the ČSN / IEC 25000 standard for a practical comparison of developed implementations. Based on a questionnaire survey, the results of individual solutions are analyzed.

Keywords: Object database, Relation database, Database design, CLASS diagram, Entity-Relationship diagram

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní době existuje mnoho lidí, kteří používají zařízení (počítač, laptop, mobil…) pro propojení s virtuálním světem. Práce ve virtuálním světě může být rozmanitá od hledání věcí na internetových stránkách, práce s textem a s daty pomocí aplikací až po hraní her. V každé činnosti se práce s počítačem nějakým způsobem dotýká databázových systémů. Internetové obchody jsou příkladem pro užití databázových technologií na internetových stránkách, kdy v databázích jsou uchovávány vlastnosti o produktu, dostupnosti, přihlašovací údaje a další údaje. Ve hrách je zase nejčastější použití databáze při uložení aktuálního stavu hry a nastavení hry (rozlišení, detaily…).

Účelem diplomové práce bude seznámit čtenáře s relační a objektovou databází. Jedná se o dvě nejpoužívanější databáze.

Teoretická část se bude zabývat CLASS diagramem a Entity-Relationship diagramem. Oba diagramy patří pod skupinu standartu Unified Modeling Language (UML). V druhé části budou vysvětleny databázové modely. Největší zřetel bude kladen na objektovou a relační databázi, kde budou vysvětleny jednotlivé trendy databází. Objektová a relační databáze bude vysvětlena pomocí definic a principů. Teoretická část bude zhotovena za účelem zjištění teoretických poznatků, které budou použity pro zhotovení praktického příkladu.

Cílem praktické části bude pomocí praktického příkladu porovnány a prozkoumány rozdíly mezi objektovou a relační databází z pohledu praktické použitelnosti. Praktická část bude zhotovena pomocí poznatků z teoretických částí. Téma praktické části bude vytvoření databáze hokejového týmu ve dvou implementacích (pro objektovou a relační databázi). Funkčnost obou implementací bude stejná. Jediný rozdíl bude spočívat v rozdílné implementaci. Pro přehlednost bude v prvé řadě vytvořen UML CLASS diagram, který bude představovat schéma pro databáze. Pro přehlednost relační databáze, kde budou vypisovány vazební tabulky, bude vytvořen implementační Entity-Relationship diagram, který graficky znázorňuje relační databázi. Následně budou pro demonstraci rozdílnosti databází zobrazeny dotazy.

Pro zhodnocení databází bude zhotoven dotazník. Otázky v dotazníku vychází z normy ČSN ISO/IEC 25000. Dotazník bude následně vyhodnocen. Výsledky budou zdůvodněny dle výsledků dotazníku. Otázky z dotazníku budou vycházet z databází,

která budou znázorněna v praktické části diplomové práce. Výsledky budou zpracovány a popsány ve výsledku práce.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je pomocí praktického příkladu porovnat relační a objektovou (grafovou) databázovou technologii z pohledu praktické použitelnosti. Práce má také za úkol seznámit s moderními trendy v databázové technologii a prozkoumat rozdíly mezi nimi.

2.2 Metodika

V první části práce bude teoretický přehled použitých zdrojů ve formě literární rešerše. Druhá část práce bude praktická ve formě projektové dokumentace databázové aplikace v oblasti sportu. Konkrétně se jedná o projektovou dokumentaci v hokejovém týmu. Na první společnou analytickou část podpořenou modely ve standardu UML budou navazovat dvě různé implementace za účelem praktického porovnání konkrétní relační a konkrétní objektové (alias grafové) databázové technologie. Tato řešení budou porovnána podle alespoň tří parametrů kvality ISO/IEC 25000.

3 Teoretická východiska

3.1 Modelování dat

Pomocí modelování dat jsou návrháři databázových systémů a informačních systémů schopni navrhnout budoucí modely. V praxi se modelování dat příliš nevyskytuje a systémy se vytvářejí při pochodu, což může návrhářům dost ztížit práci a nemají faktický přehled o skutečném rozložení databáze. Modelování dat může být ale velice užitečné, protože vývojář díky němu má větší přehled o funkčnosti a o jednotlivých prvkách systému.

V kapitole Modelování dat jsou vysvětleny základní principy a rozdělení při modelování dat. Po popsání všeobecných principů modelování dat jsou blíže představeny diagramy ER a diagram tříd (CLASS diagram).

3.1.1 Druhy přístupů

V této kapitole jsou popsány základní principy jednotlivých druhů přístupů. Druhy přístupů jsou popsány od nejstaršího po nejnovější.

- **Funkční přístup**

  Dle (Kaluža a Kalužová 2011) je funkční přístup jeden z nejstarších způsobů pro modelování dat. Vychází z předpokladu, že v první fázi se provede určení jednotlivých funkcí (procesy, činnosti…). Po první části přichází na řadu propojení jednotlivých funkcí pomocí toků, které slouží k specifikaci mezi jednotlivými funkcemi.

- **Datový přístup**

  Autoři (Kaluža a Kalužová 2011) popisují rozdíl mezi datovým a funkčním přístupem. Rozdíl spočívá v tom, že datový přístup v prvním kroku vytvoří celkový datový model. Druhý krok spočívá ve vypsání jednotlivých funkcí. Mezi nejdůležitější událost pro vývoj datového modelu lze započítat vývoj databázových systémů.

  Autoři (Kaluža a Kalužová 2011) považují jako hlavní vývoj datového přístupu ER-diagram, díky kterému lze systém převést do grafické podoby. Jeden z nejznámějších autorů ER diagramů je Peter Chen.

- **Objektový přístup**

Autor (Dařena 2004) popisuje objektový přístup jako jeden z nejmodernějších druhů přístupů k modelování dat. Objektový přístup vznikl na konci 70. let 20. století. Objektový přístup je založen na principu, že základem přístupu je objekt. Rozdíl oproti předchozím přístupům (funkční a datový přístup) spočívá ve vzniku diagramu.

Autor internetového článku (Dařena 2004) vysvětluje pojem objektový přístup. Objektový přístup je založen na principu, že nejdříve jsou vytvořeny základní stavební prvky, které jsou následně složeny dohromady v celý program. Tento princip vytváření se nazývá metoda zdola nahoru. Výhoda této metody spočívá v tom, že jednotlivé prvky se dají opětovně použít. Podmínka pro opětovné použití spočívá v zajištění komunikace s ostatními prvky. Rozdíl mezi jednotlivými objekty v jednom programu jsou ve vlastnostech. Typy vlastností ve třídě objektu se nazývají atributy. V každé třídě musí být jeden atribut, který bude přesně odlišovat každý jednotlivý záznam. Hlavní princip objektového přístupu spočívá v objektech. Systém potom obsahuje několik objektů, kde každý objekt obsahuje datovou strukturu a metody. Jednotlivé objekty pak dokážou mezi sebou „komunikovat“, což způsobí, že k jednotlivým objektům je vyvolána metoda.

3.1.2 Sémantický datový model

Dle autorů knihy (Kaluža a Kalužová 2011) je sémantický datový model určen pro nalezení typů objektů, které budou nejlépe vyobrazovat modelovanou realitu. Postup v sémantickém datovém modelu závisí na stanovení cíle, identifikace datových prvků (charakteristik) a finální verzi modelu. Hlavní cíl je oproti konceptuálnímu datovému modelu v tom, že sémantické modelování dat má za cíl ukázat přímý odraz reality, a ne vytvořit představu celkového modelu. Úkol sémantického modelu je úvaha o výsledné podobě modelu.

3.1.2.1 Cíl

Internetová stránka (informacni-systemy.studentske 2021) vysvětluje cíl sémantického datového modelu. Mezi hlavní cíl sémantického datového modelu je nalezení datové části, která je vybrána pomocí přesně daných specifik. Datové části se rozdělují na dva druhy: hmotné a nehmotné. Mezi nehmotné části lze jako příklad uvést entity Faktura a Objednávka. Mezi hmotné části lze pak zařadit entity Student, Profesor a Auto.

3.1.2.2 Návrh metodiky sémantického modelování

Sémantické datové modelování má dle (Kaluža a Kalužová 2011) tři kroky k vytvoření modelu. Kroky v návrhu jsou identifikace vstupních datových požadavků, specifikování (charakterizování) objektů a vztahů. Poslední krok je revize struktury objektů.

1. Identifikace vstupních datových požadavků

   V první řadě je důležité definice cíle a rozsahu práce. Tyto definice jsou odvozeny ze vstupních požadavků, které má daný projekt zadán. Správná definice je důležitá z pohledu, že v jednom projektu mohou být data nevýznamná, ale v jiném hrají velmi důležitou roli. K vymezení cíle se používají nejčastěji metody, jako jsou například dotazníkové šetření a pozorování. Pro správné pojmenování cíle se používá dokumentace, která obsahuje slovní popisy, vstupní/výstupní formuláře a formáty stávajících struktur.

2. Specifikování a charakterizování objektů (vztahů)

   Účelem tohoto kroku je správné specifikování a charakterizování objektů a jejich vztahů. K popsání objektu může být použito slovní popsání pomocí schématu, kdy se určí název objektu, popis a charakteristiky. Tímto stylem lze popsat i jednotlivé vztahy, které se vyskytují mezi objekty.

3. Revize struktury typů objektů

   Revize obsahuje srovnávací analýzu prvků, které identifikují negativní rysy modelu. Chyby mohou obsahovat podobné názvy objektů a charakteristik, redundance typů objektů a prvky objektivní reality, které obsahují stejné prvky.

3.1.3 Konceptuální datový model

Kniha od autorů (Chlápek, Ph.D., Kučera, Ph.D. a Ph.D. Palovská 2019) vysvětluje základ konceptuálního datového modelu pomocí grafického znázornění řešeného problému. Jako první diagram v konceptuálním datovém modelu je ER diagram, z kterého se postupně vyvinul diagram tříd. Hlavní účel konceptuálního datového modelu je vytvořit entity, atributy a vztahy s kterými lze dále pracovat. V současné době patří mezi nejpoužívanější datové modely E-R diagram (3.1.4) a diagram tříd (3.1.5).

Autoři (Chlápek, Ph.D., Kučera, Ph.D. a Ph.D. Palovská 2019) popisují výhodu v konceptuální úrovni v možnosti nezatížit databázový a informační systém. Konceptuální úroveň se zabývá návrhem zkoumaného problému. Uživatel může pomocí tohoto modelu vytvořit databázi nebo informační systém. Konceptuální datový model je rozdělen na konceptuální schéma a konceptuální model dat.

- Konceptuální schéma

  Konceptuální schéma je vyznačeno modelem, který je představen za účelem analýzy problematiky návrhu dat. Konceptuální schéma obsahuje základní entitní množiny a vztahy, které jsou mezi entity a atributy.

- Konceptuální model dat

  Konceptuální model dat spočívá v detailním specifikování navrhované databáze. Konceptuální model dat zahrnuje všechny entitní množiny, atributy, datové typy a vztahy. Vztahy v modelu dat mohou již obsahovat rozmezí M:N.

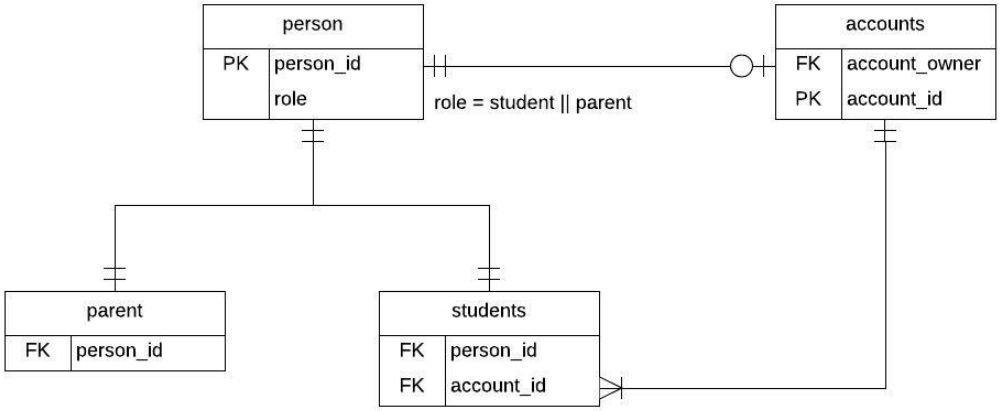
3.1.4 Entity-Relationship diagram

Entity-Relationship diagram je také znám pod zkratkou E-R diagram. E-R diagram je použit v praktické části práce pro návrh relačního databázového systému. E-R diagram je vytvořen v praktické části implementačně z důvodu znázornění vazebních tabulek mezi třídami (kapitola 4.3.1). Datové typy v E-R diagramu a pro relační databázi jsou stejné, proto v relační databázi nebudou znázorněna.

3.1.4.1 Základní informace E-R diagram

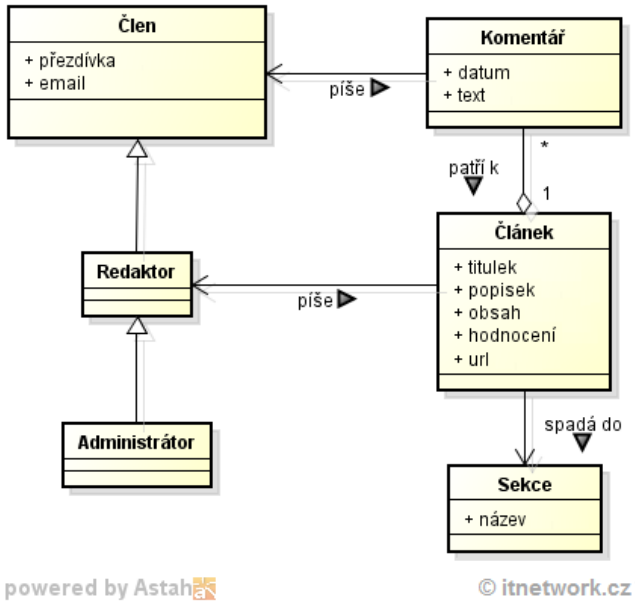
Autor (Peterson, guru99 2021) považuje E-R diagram jako jeden z nejstarších a také nejúspěšnějších konceptuálních datových modelů. E-R diagram byl vytvořen v roce 1976 Petrem Chenem. Zásadním průlomem byla publikace článků, které vyšly pod názvem Model vztahů mezi entitami: Směrem k jednotnému pohledu na data.

(Peterson, guru99 2021) představuje jako předchůdce E-R diagramu Bachmanův diagram, který byl vytvořen Charlesem Bachmanem mezi 60. a 70. lety 20. století. Postupem času se vyvíjel a v roce 1988 byl přijat americkým institutem ANSI jako standart pro konceptuální datový model. Specifikum pro diagram je v použití termínů entita, vztah a multiplicita.



Obrázek 1-E-R diagram pomocí notace Crow's foot, Zdroj: (stackoverflow 2021)

Autoři (Kaluža a Kalužová 2011) považují jako hlavní součást E-R diagramů soustavu konstruktorů (prvků), které je při vývoji potřeba správně definovat. E-R diagram slouží ke grafickému znázornění jednotlivých prvků a jejich vztahů. Grafické znázornění se odlišuje dle softwaru, který je pro vytvoření použit. Základní komponenty pro E-R diagram jsou Entita, Atribut a Vztah.



Obrázek 2- E-R diagram, zdroj: (itnetwork 2021)

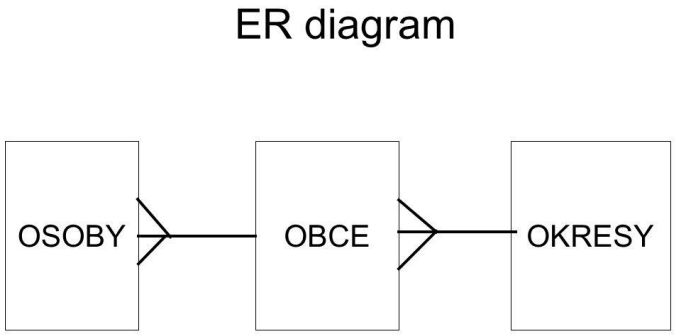
3.1.4.2 Základní konstruktory E-R diagramu

Internetová stránka (vovcr 2021) zařazuje mezi základní konstruktory E-R diagramu entitu, atribut, doménu, relaci (kardinalita a parcialita) a klíče (primární, cizí…).

- Entita

  (it-slovnik.cz 2021) vysvětluje pojem entita jako vyjádření pro jakýkoli objekt, který slouží jako popis reálného světa. Další podmínkou je použití dané entity v datovém modelu. Entita musí být specifikována tak, že není závislá na ostatních entitách.

  (Kaluža a Kalužová 2011) vysvětlují příklad pro entitu objekt Zaměstnanec, Zvíře a Výrobek. Grafické znázornění entity je v podobě obdélníku, kde je jméno uvedeno v horní části. Označení pro entitu by mělo být jednoznačné a přesně definované.

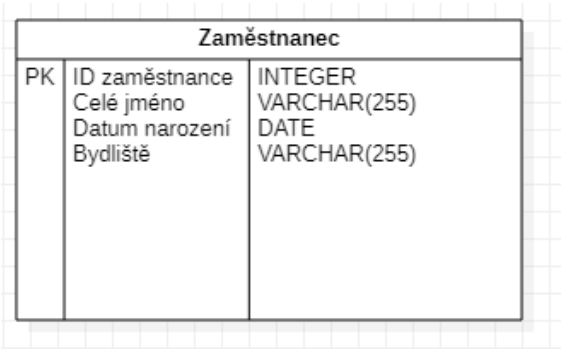


Obrázek 3- Entita, Zdroj: (slideplayer 2021)

- Atribut

Atribut je dle autora z internetové stránky (Peterson, guru99 2021) určen k popisu vlastností entit. Příkladem je entita Zaměstnanec, který obsahuje atributy Jméno, Příjmení, Rok narození a Pozice. Každý atribut má vlastnost dosahovat předem stanovených hodnot. Atribut se dělí na povinný a nepovinný. Rozdíl je v tom, že povinný atribut musí mít zapsanou hodnotu. Nepovinný atribut může dosahovat tzv. hodnoty NULL, která se popisuje níže v atributu. V grafické podobě jsou atributy znázorněny v dolní části značky entity. Atribut se dále dělí na jednoduchý, složený, odvozený a vícehodnotový.

- Jednoduchý atribut – Nelze dále dělit (Kontaktní číslo, ID)
- Složený atribut – Lze rozložit (Celé jméno studenta – rozdělí se na Jméno a Příjmení)
- Odvozený atribut – Není uložen přímo v databázi, lze dopočítat (věk z roku narození)
- Vícehodnotový atribut – Více než jedna hodnota (telefonní číslo, půjčené knihy).



Obrázek 4- Atributy, Zdroj: Vlastní zpracování

Dle (visual-paradigm 2021) atribut obsahuje model přípustných hodnot, které může atribut nabývat. Datový typ slouží jako popis vlastností (charakterizování) atributu. Použití atributu entit se liší dle jednotlivých situací a podle požadavku na entitu. Atributu entity se někdy též říká datový typ. Datových typů je velké množství, kde každý má své výhody a nevýhody (použití). Datové typy lze rozdělit do několika skupin. Přiřazené hodnoty mohou být pro jeden či více atributů. Příkladem pro datový model je entita Pes, která má atribut Plemeno, které je označeno datovým typem STRING a má omezení velikosti na 12 míst.

Autor (Čápka, itnetwork 2021) rozdělil datové typy na celá čísla (Tabulka 1- Datové typy pro celá čísla, Zdroj: , desetinná čísla (Tabulka 2), texty (Tabulka 3- Datové typy pro texty, Zdroj: , hodnotu NULL a ostatní (Tabulka 4).

- Celá čísla

Datový typ	Rozsah	Velikost (bit)
INTEGER (INT)	-2 147 483 648 až 2 147 483 647	32 bitů
Byte	0 až 255	8 bitů
Sbyte	-128 až 127	8 bitů
ushort	0 až 65 535	16 bitů
short	-32 768 až 32 767	16 bitů

Tabulka 1- Datové typy pro celá čísla, Zdroj: (itnetwork 2022)

- Desetinná čísla

Datový typ	Rozsah	Přesnost
Float	+(-) 1,5 *10^{45} až +- 3,4 * 10^{308}	7 čísel
Double	+-5.0 * 10^{-324} až +-1.7 * 10^{308}	15-16 čísel

Tabulka 2- Datové typy pro desetinná čísla, Zdroj: Zdroj: (itnetwork 2022)

- Texty

Typ	Rozsah
TINYTEXT	Maximálně 255 B
TEXT	Maximálně 64KB (články)
MEDIUMTEXT	Maximálně 2^{24} B
LONGTEXT	Maximálně 2^{32} B
VARCHAR (max. počet znaků)	Počet zvolených znaků (max 64KB)
CHAR (počet znaků)	Pevný počet znaků (předem zvolený) max 255

Tabulka 3- Datové typy pro texty, Zdroj: (itnetwork 2022)

- Ostatní

Typ	Rozsah
DATE	Textový řetězec ve tvaru rrrr-mm-dd (rok, měsíc, den) Používá se z důvodu popsání datumu
TIME	Čas jako textový řetězec hh:mm:ss (hodiny, minuty, sekundy)
BOOL	TRUE nebo FALSE (8 bitů) Když je požadován výstup ano nebo ne.

Tabulka 4- Ostatní datové typy, Zdroj: (itnetwork 2022)

- Hodnota NULL

Hodnota NULL v E-R diagramu neznamená hodnotu nula. Hodnota NULL označuje skutečnost, že hodnota není zadána. Prázdná hodnota se tudíž nastaví pomocí výchozí hodnoty na NULL v SQL, pokud se tato hodnota nenastaví explicitně jinak. Lze zadat i název, který se pojmenuje jako NOT NULL, to znamená, že v tomto atributu se musí při každém novém záznamu zadat

hodnota (nemůže být prázdné pole). Hodnota NULL se používá nejčastěji při definování primárního klíče.

- Relace

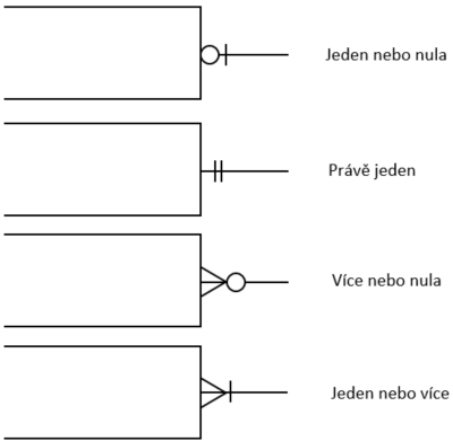
Kniha od (Kaluža a Kalužová 2011) vysvětluje relaci. Základní princip relace je rozeznání, jaký vztah souvisí mezi dvěma entitami. Relace je označena pomocí slovesa mezi dvěma entitami, které jsou označeny podstatným jménem. Jako příklad pro relaci lze uvést entity Zákazník a Prodavač. Tyto dvě entity lze spojit pomocí relace Prodává.

Obrázek 5-Relace u Entity relationship diagramu, Zdroj: (vovcr 2022)

- Kardinalita

Internetová stránka (sallyx 2015) představuje kardinalitu pro E-R diagram, která může nabývat třech hodnot. První možnost je 1:1, druhá možnost je 1:M a třetí M:N.

- Vztah 1:1 představuje možnost, kdy jedné entitě A může existovat pouze jedna entita B a naopak. Jako příklad vztahu 1:1 lze uvést entity Zaměstnanec - Čipová karta.
- 1:M značí, že jedné entitě A může odpovídat více jak jedna entita B. Jako příklad lze uvést entity Zákazník - Objednávka. Naopak entitě B odpovídá pouze jedna entita A.
- M:N značí, že entitě A může odpovídat jedna či více entit. Tento vztah slouží i obráceně. Jako příklad vztahu M:N lze uvést entity Studenti-Učebny.



Obrázek 6-Kardinalita u E-R modelu, Zdroj: (vovc 2022)

- Particialita

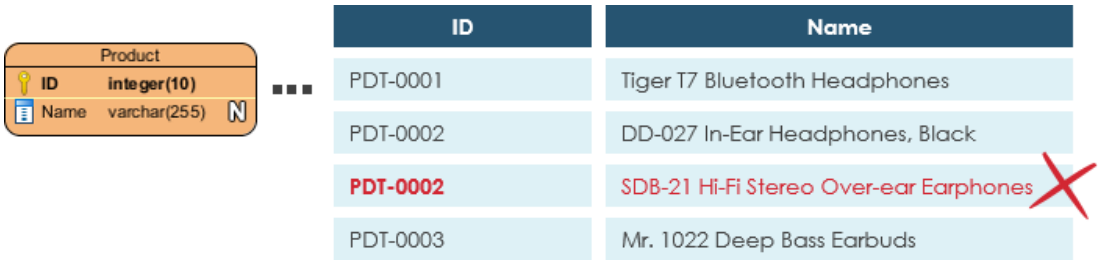
Autor (Zechmeister 2021) vysvětluje parcialitu. Parcialita vztahu určuje, jedna entita může existovat bez instance na druhé straně. Parcialita se rozděluje na dvě možnosti 0..1 a 0..M. Příklad pro 1:0..(M) je Zákazník - Objednávka. Existence zákazníka není závislá na objednávce, ale objednávka nemůže existovat bez zákazníka. Parcialita může mít možnost, že na jedné straně (obou) bude například 0*..M, což určuje, že entita může fungovat bez existující instance na druhé straně.

- Primární klíč

Web (visual-paradigm 2021) vysvětluje primární klíč. Primární klíč v E-R je znázorněn z důvodu jednoznačné identifikace tabulky. Platí, že v jedné entitě ER diagramu nemůžou být dva stejné atributy se stejnou hodnotou, které jsou označené primárním klíčem. V E-R diagramu je primární klíč označen pomocí zkratky PK (Primary Key). Primární klíč se musí vytvářet vždy s rozvahou, protože to musí býti informace, která nesouvisí s ochranou osobních údajů a která rozlišuje jednotlivé atributy. Příkladem je firma, která použije jako primární klíč Osobní číslo zaměstnance.

Doporučení od Connolly & spol pro primární klíč (2008) (Kaluža a Kalužová 2011)

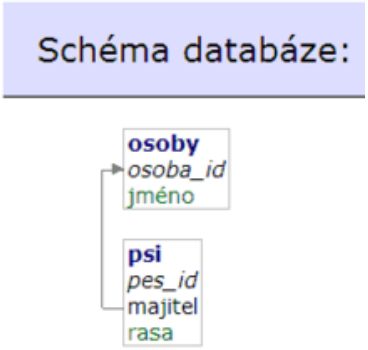
- Minimální množina atributů
- Malá pravděpodobnost změny a ztracení jednoznačnosti
- Malý počet znaků (text) a malá maximální hodnota (číslice)



Obrázek 7- Primární klíč v E-R diagramu. Zdroj: (visual-paradigm 2022)

- Cizí klíč

Dle (visual-paradigm 2021) cizí klíč slouží k rozeznání atributů, které slouží k propojení a identifikaci mezi jednotlivými entitami. Cizí klíč je značen zkratkou FK (Foreign key). Jelikož cizí klíč je rozdílný od primárního klíče, tak nemusí být jedinečný. V jedné entitě může být více cizích klíčů se stejným atributem. Příklad pro cizí klíč jsou entita Člověk a Barák. Entita Člověk obsahuje tři záznamy se stejnou adresou označenou jako cizí klíč. V druhé entitě Barák je primární klíč Adresa, která je propojena s entitou Člověk právě přes Adresu.



Obrázek 8- Cizí Klíč v E-R diagramu, Zdroj: (upload.wikimedia 2021)

3.1.4.3 Použití E-R diagramu

Webová stránka (lucidchart 2021) vypisuje, v jakých oborech se používá E-R diagram. Mezi příklad použití E-R diagramu lze zařadit práce s databází (návrh a odstranění problému), informační systémy a výzkum.

- Návrh databáze

Dle (lucidchart 2021) je použití E-R diagramu při návrhu databáze. V případě návrhu databáze se používá E-R diagram pro relační databázový přístup z hlediska modelování

a návrhu pro konkrétní databázi. Dále je vhodný z hlediska logiky (logický datový model) a specifické technologie, která má být implementována. Další použití E-R diagramu je v softwarovém inženýrství, kde se používá pro návrh informačních systémů.

- Odstranění problémů

Dle (lucidchart 2021) je použití E-R diagramu pro odstranění problémů. V případě, že v relační databázi nastane nějaký problém, lze použít E-R diagram bez fyzického opravení databáze. Při nakreslení uživatelé diagramu mají možnost přehlednější správy fyzické databáze, a tak lze snadněji nalézt chybu. Pomocí diagramu lze odhalit, kde se v databázi chyba stala.

- Informační systémy

Dle (lucidchart 2021) lze E-R diagram použít v informačních systémech. Jedná se o jakékoliv informační systémy, které obsahují entity.

- Výzkum

Databáze se v dnešní době používají pro relační informace. V rámci vzdělání je to užitečné z pohledu následného vyhledávání. (lucidchart 2021)

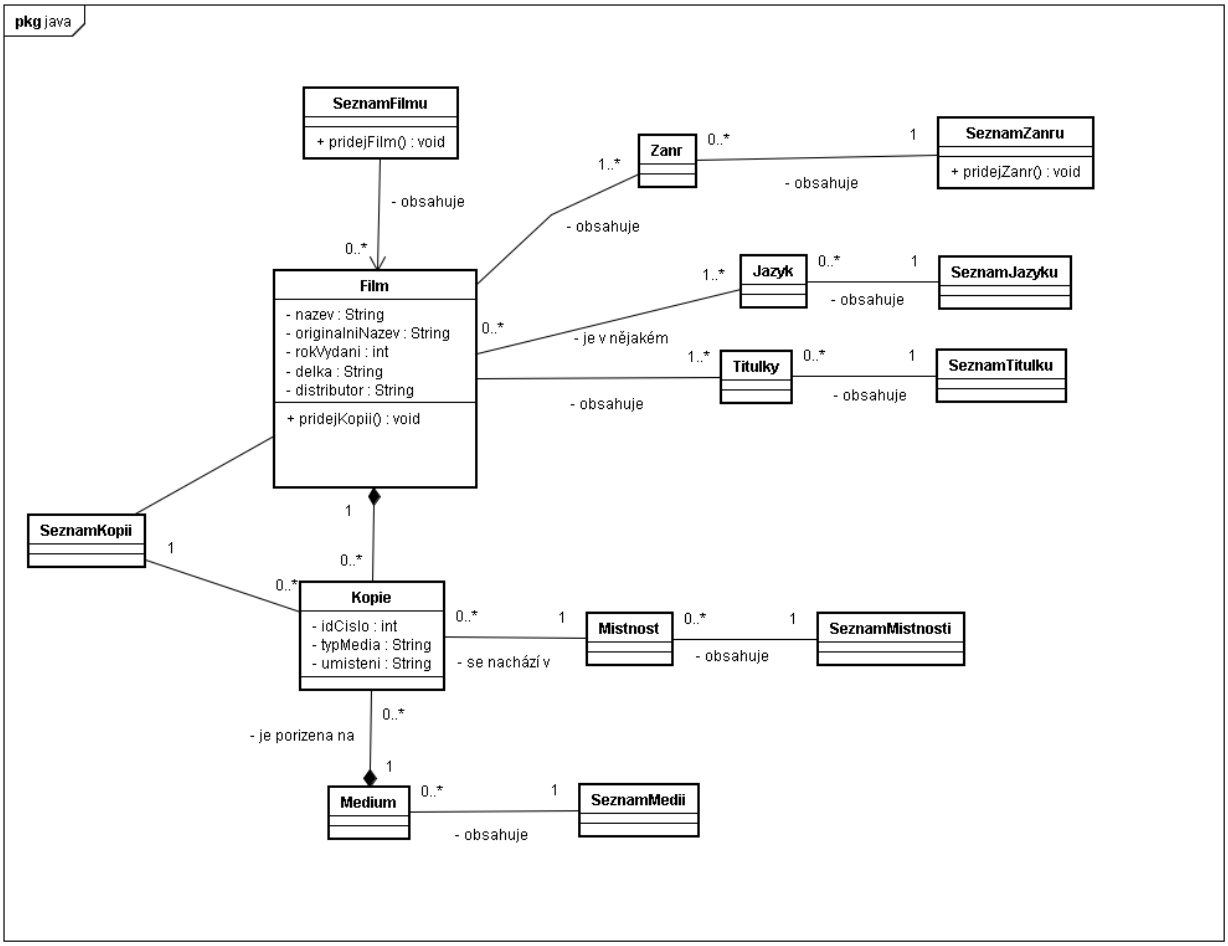
3.1.5 Diagram tříd

Diagram tříd lze přeložit pod pojmem CLASS diagram. Oba názvy se v českém prostředí vyskytují velmi často. V diplomové práci je diagram tříd uveden z důvodu obecného návrhu UML, z kterého následně vychází objektový a relační databázový model. Návrh diagramu tříd je znázorněn pro praktickou část (kapitola 4.2).

Webová stránka (tutorialspoin 2021) vysvětluje Diagram tříd jako grafický zápis, který lze použít při konstrukci a vizualizaci objektu v systému. Diagram tříd je jeden z nejčastěji používaných diagramů pro objektově orientované systémy. Diagram tříd je statická konstrukce znázorněna v softwarovém inženýrství, která je schopna poskytnou přehled o systému tím, že dokáže zobrazit třídy, atributy, operace a vztahy mezi třídami. Diagram tříd se používá nejčastěji při vizualizaci (popisu) vybraných aspektů systémů a při konstrukci kódu. CLASS diagram lze použít jak pro celý systém, tak i na jeho části.

Dle internetové stránky (ecom.ef.jcu 2022) Diagram tříd znázorňuje vzájemné interakce. Diagram tříd neznázorňuje, co se při interakcích děje. To znamená, že diagram

tříd zobrazuje statický stav. Mezi třídami se znázorňují vztahy (vazby), které mohou být vícenásobné.



Obrázek 9- Příklad diagramu tříd, Zdroj: (java.vse 2021)

Pro třídu lze na internetu nalézt mnoho definic. Jako příklad lze uvést definici Vondrák (2004) „*Třída je popis objektů mající společnou strukturu, chování, vztahy a sémantiku*“. Jednotlivé objekty ve třídě se nazývají instance. Třídu lze definovat výčtem vlastností a výčtem objektů. (Szturcová 2022)

- Výčet vlastností

  (Szturcová 2022) popisuje výčet vlastností jako extenzivní definování. Princip spočívá ve vypsání všech vlastností, které jsou přiřazeny k danému objektu. Jako příklad lze uvést třída Kočka, která v extenzivním definování bude mít vlastnosti: savec, 4 nohy, hodně spí…

- Výčet objektů

(Szturcová 2022) popisuje druhou metodu definice třídy. Výčet vlastností je v intenzivním pojetí. To znamená, že třída představuje jednotlivé objekty. Intenzivní pojetí spočívá v konkrétních příkladech pro výčet vlastností. Jako příklad lze uvést třída Kočka, kde jsou jména jednotlivých koček, jednotlivá barva…

Internetová stránka (docwiki.embarcadero 2021) popisuje Diagram tříd. Diagram tříd je schopen popsat typy objektů a vazby, které jsou mezi jednotlivými objekty. Diagram tříd je schopen zachytit vlastnosti, operace a omezení v jednotlivých objektech. Specifikace diagramu tříd je v zobrazení přehledu systému. Diagram tříd je schopen zobrazovat třídy, rozhraní a vztahy mezi jednotlivými objekty. Diagram tříd zobrazuje, co se integruje, nikoliv co se děje během integrace, a proto se říká, že diagram tříd je statický. Třída má možnost odkazovat na jinou třídu a dědit z jiné třídy.

Dle (docwiki.embarcadero 2021) zobrazení UML třídy je v podobě obdélníku, který je rozdělen na několik částí. Názvy částí jsou název třídy, atributy a operace (metody). Ve vrchní části třídy je znázorněn název třídy. Prostřední část obsahuje jednotlivé atributy třídy a poslední část metody. Prostřední a poslední část znázorňuje jednotlivé části, které patří k dané třídě.

3.1.5.1 Jméno třídy

Internetová stránka (docwiki.embarcadero 2021) definuje název třídy. Název třídy se zapisuje v prvním oddílu diagramu tříd. Název třídy má popisovat účel, pro který je daná třída vytvořena. Pro jméno třídy je specifické označení pomocí tučného písma a na střed buňky. Každé jméno třídy musí být jedinečné, aby se v diagramu dalo bezpečně rozeznat a identifikovat. Při návrhu jména třídy se doporučuje postupovat:

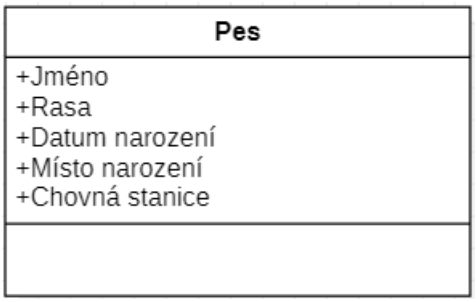
1. Jméno třídy by mělo začínat velkým písmenem.
2. Jméno třídy by mělo být uprostřed buňky.
3. Jméno třídy by mělo být napsáno tučně.
4. Do jména třídy se nedávají názvy činností, pokud je to možné.
5. Do jména třídy nedávat vlastnosti třídy.

Obrázek 10- Jméno třídy v CLASS diagramu, Zdroj: Vlastní zpracování

3.1.5.2 Atribut třídy

Autorka (Szturcová 2022) popisuje atribut třídy jako vlastnosti, které se používají v dané třídě. Atribut třídy se používá k popisu informací třídy, ve kterém jsou zapsané. Atribut třídy je zapsán v druhém oddílu třídy. Existuje více možností zápisu atributu třídy v CLASS diagramu. Příkladem pro zápis atributu je syntaxe: [viditelnost] [/] název [:typ] [násobnost].

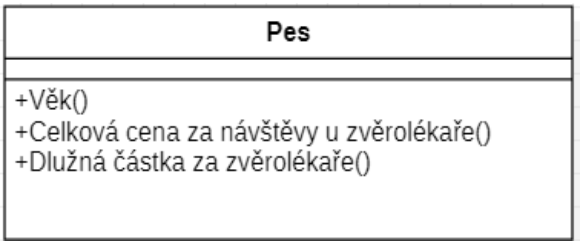
Dle (cw.fel.cvutv 2022) atribut třídy může existovat bez toho, aniž by existoval jediný záznam (instance) dané třídy. Atribut slouží jako vlastnost třídy v celkovém pojetí, a ne jako vlastnost jednotlivých instancí ve třídě. Jednotlivé instance v atributu mohou mít rozdílnou hodnotu (textový řetězec, numerická hodnota …).



Obrázek 11- Atribut třídy v CLASS diagramu, Zdroj: Vlastní zpracování

3.1.5.3 Operace tříd

Autorka (Szturcová 2022) popisuje operace tříd. Operace se používá k definování chování a služeb, které objekt (třída) poskytuje. K spuštění chování je zapotřebí poskytnout všechny informace. Tyto informace jsou poskytnuty pomocí operace tříd. Syntaxe pro operaci je [viditelnost] název ([seznam_parametrů]) ':' [návratová hodnota].



Obrázek 12- Operace v CLASS diagramu, Zdroj: Vlastní zpracování

3.1.5.4 Viditelnost operací a atributů

Autor (Čápka, itnetwork 2022) popisuje základní typy viditelnosti pro atributy a operace. Existují 4 typy viditelnosti pro operace a atributy. Typy viditelnosti jsou soukromé atributy, veřejné atributy, chráněné atributy a v rámci balíčku.

- Soukromý atribut (Private)

  Privátní atribut slouží k zobrazení v rámci dané třídy. Pracovat s atributem, který má typ Private, mohou pouze operace v dané třídě. Značení pro privátní atribut je znak mínus.

- Veřejný atribut (Public)

  Veřejný atribut je vidět v rámci celého systému. S veřejným atributem se může pracovat skrze jiné třídy a řešit metody, což je hlavní rozdíl mezi veřejným atributem a privátním atributem. Značení pro veřejný atribut je pomocí znaku plus (+).

- Chráněný atribut (Protected)

  Chráněný atribut je podobný jako soukromý. Rozdíl spočívá v tom, že viditelnost v soukromém atributu je pouze v rámci té třídy. V chráněném atributu je viditelnost v dané třídě a u všech potomků, které jsou součástí třídy. Značení pro chráněný atribut je pomocí znaku hashe (#).

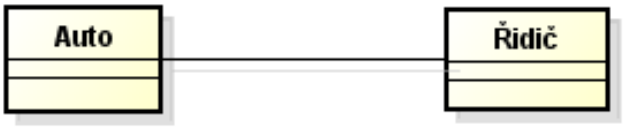
- Atribut viditelný v rámci balíčku (Package)

  Atribut viditelný v rámci balíčku je viditelný pouze v předem daném balíku. V balíku jsou předem předdefinované třídy. Balík je uspořádán do logických celků, kde jsou uvnitř předem definované třídy. Znak pro atribut v rámci balíčku je tilda (~).

3.1.5.5 Vztahy mezi třídami

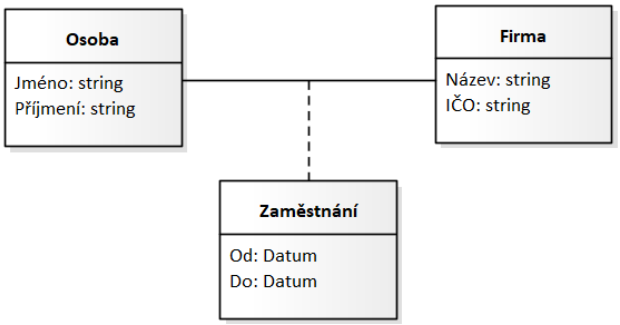
- Asociace

Dle (Čapka 2021)je asociace nejjednodušší vztah mezi entitami. Entity v asociaci mohou záviset nezávisle na druhé entitě. Jako příklad lze uvést Auto a Řidič, kde jednotlivé entity nejsou závislé na druhé. Asociace se značí plnou čarou. V plné čáře je určení, že entity mají odkaz jedna na druhou. Pomocí šipky k jedné entitě lze určit, že informace si uchovává pouze ta entita, ze které šipka směřuje. Sémantický význam vazby ovlivňuje název asociace. Asociace se většinou označuje slovesem anebo slovesným spojením.



Obrázek 13- Binární asociace, Zdroj: (itnetwork 2021)

Autor (Rydval 2021) popisuje asociační třídu, kterou je možné přidat do asociace. O asociační třídě platí, že se jedná o třídu i asociaci v jedné chvíli. Asociační třída se používá v případech, když je potřeba uložit vlastnosti, které se nehodí ani do jedné třídy.



Obrázek 14- Asociační třída, Zdroj: (modelovaci-jazyky 2021)

- Agregace

Autor (Čapka 2021) vysvětluje agregaci jako vztah mezi dvěma entitami. Tento vztah určuje, že entity se rozdělí na celek a část. Mezi příklad agregace lze uvést objekty Sekce a Článek. Článek obsahuje sekce, tudíž je jasné, že mezi těmito dvěma objekty existuje agregace. Článek může obsahovat 1 až nekonečno sekcí, označeno 1...*.

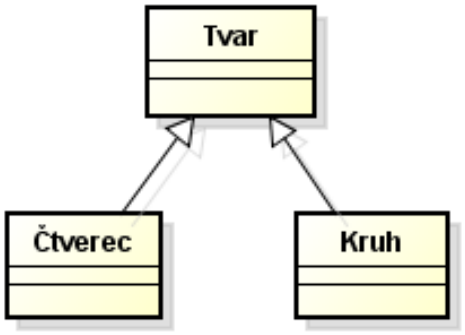
Pro zakreslení se používá plná čára zakončená prázdným kosočtvercem. Kosočtverec určuje objekt, který reprezentuje celek (Obrázek 15-Agregace). Takže jednotlivé části mohou existovat bez celku a také části mohou být obsaženy ve více celkách. V agregaci také může nastat, že jedna strana bude mít možnost 0…*, to je že entita může existovat bez té druhé.

Obrázek 15-Agregace, Zdroj: (itnetwork 2021)

- Generalizace

(Kaluža a Kalužová 2011) definují generalizaci jako vztah mezi dvěma či více objekty. V generalizaci je jedna třída. Jeden (či více) tříd jsou potomci. Účelem potomků v generalizaci je, že dědí všechny informace od rodiče neboli potomci dědí vlastnosti a chování rodiče. Použití generalizace je vhodné v případě, kdy existuje více tříd se stejnými atributy. Zde lze vytvořit jednoho rodiče, kde jeho potomci (podtřídy) budou dědit právě atributy, které jsou uvedeny v nadtřídě. Generalizace má schopnost mít hierarchii, kde může být libovolný počet asociací (do hloubky i šířky).

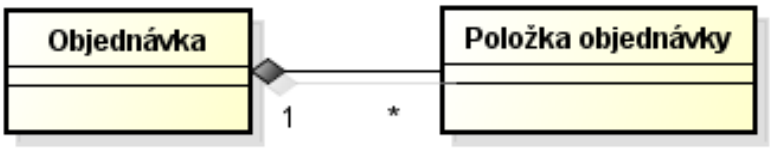
Internetová stránka (docplayer. 2022) dělí Generalizaci na jednoduchou dědičnost a vícenásobnou dědičnost. Jednoduchá dědičnost spočívá v podtřídě (potomku), který dědí atributy pouze od jedné nadtřídy (rodiče). Vícenásobná dědičnost spočívá v tom, že podtřída (potomek) má více nadtříd (rodičů). Ve vícenásobné dědičnosti existuje pravidlo, že nadtřídy by měly být na sebe nezávislé.



Obrázek 16- Generalizace, Zdroj: (itnetwork 2021)

- Kompozice

Kompozice dle (Čapka 2021) je vztah mezi dvěma entitami, které se blíží agregaci. Rozdílem je, že v případě kompozice je určen silnější vztah, neboli by entita části neměla smysl bez celku. V případě, že zanikne celek, tak zaniknou i jejich kompoziční prvky (části). Kompozice se zakresluje podobně jako agregace. Jediný rozdíl spočívá v kosočtverci, který je plný. Dále celek je vždy označen číslem jedna, což znamená, že jeden celek obsahuje určitý počet částí.



Obrázek 17- Kompozice, Zdroj: (itnetwork 2021)

- Multiplicita

Multiplicita dle (Kaluža a Kalužová 2011) označuje mnohonásobnou vazbu mezi objekty. Multiplicita lze uvést ve vazbách, které obsahují asociaci, agregaci a kompozici. Multiplicitu lze napsat třemi způsoby. Multiplicita se používá mezi dvěma třídami.

1. 1 (číslo)- Označuje konkrétní hodnotu mezi entitami
2. * (hvězdička)- Označení pro libovolný interval včetně nuly
3. 1..* (hvězdička označuje interval) - Interval se značí pomocí dvou teček. Jako příklad lze uvést omezení třídy na interval od čísla 5 a více - 5...* (Kaluža a Kalužová 2011)

3.1.5.6 Využití CLASS diagramu

Dle (ibm 2022) se CLASS diagram používá k vytvoření systému nebo subsystému. CLASS diagram se používá pro modelování objektů, zobrazování vztahů mezi objekty a popisy funkčnosti objektů.

Dle (ibm 2022) CLASS diagram je používán při fázi analýzy. Diagram tříd se používá k porozumění a identifikaci požadavků systému. V objektově orientovaných softwarech se CLASS diagram používá k znázornění a identifikaci jednotlivých součástí (tříd). Tento návrh se často dále převádí do skutečných tříd a objektů. Později je možné

se vrátit k návrhu CLASS diagramu a poupravit a upřesnit počáteční požadavky. Během implementační fáze je možnost převádět model (CLASS diagram) na skutečný kód a naopak.

Diagram tříd se používá k vizualizaci a specifikaci prvků v modelech. Mezi příklady pro využití diagramu tříd patří podle (ibm 2022):

- Zachycení a definice struktury tříd
- Definice vztahů mezi třídami a klasifikátory
- Definice atributů a operací
- Definice tříd v balíčku
- Zobrazení hierarchie při dědičnosti

3.2 Databáze

(it-slovnik 2022) definuje databázi jako strukturovaný souhrn dat s kterými lze dále pracovat. K práci s daty je v databázích potřeba nějaký databázový systém. Vyhledávání, porovnávání, třídění a editování jsou příklady, které lze použít při manipulaci s daty v databázi.

Internetová stránka (quickbase 2022) říká, že v dnešní době se databáze používají pro zlepšení každodenního života. Data z databází jsou využívána například v cloudových uložištích, k předpovídání počasí, pro online nákupy. Databáze jsou v dnešní době v mnoha podobách, což si ani uživatel neuvědomuje. V dnešní době existuje mnoho společností zabývající se databázemi. Mezi nejvýznamnější lze zařadit Oracle, MySQL a DB2.

(Lutkevich 2021) definoval databázi jako softwarový systém, který vychází z anglických slov data a base, což lze přeložit jako základna pro data. Databáze je softwarový systém. Databáze je soubor, který je přizpůsoben ke snadnému přístupu, správě a aktualizaci dat. Mezi nejčastější příklady použití databází jsou uložená data, která představují prodejní transakce, zákaznická data, finance a informace o produktech. Databáze jsou určeny k shromáždění dat a informací o místech, osob a věcí. Všechny informace jsou shromážděny na jednom místě (databázi), kde je může osoba s příslušnými právy analyzovat a zpracovávat. Obecně lze databáze dělit na operační a analytické databáze.

- Operační databáze

Autoři (Hernandez a John 2004) říkají, že operační databáze jsou v současné době více používané. Tyto databáze představují „páteř“ mnoha společností (organizací). Operační databáze se vyznačují každodenní změnou dat, proto je operační databáze dynamického typu. Operační databáze vykreslují aktuální informaci o systému. Příkladem pro použití databáze je použití u obchodu, výrobní společnosti, nemocnice.

- Analytické databáze

Význam analytické databáze je dle (Hernandez a John 2004) v uložení dat v širším časovém horizontu. Uložená data jsou na sobě závislá. Informace získané v analytické databázi jsou určena k zobrazení některého okamžiku. Význam databáze je ve sledování trendů, zobrazení dlouhodobých statistických údajů a k obchodním strategiím. Příklad použití databáze je v chemických laboratořích, u geologické společnosti a u firmy používající marketingové analýzy.

3.2.1 Typy databází podle struktury dat

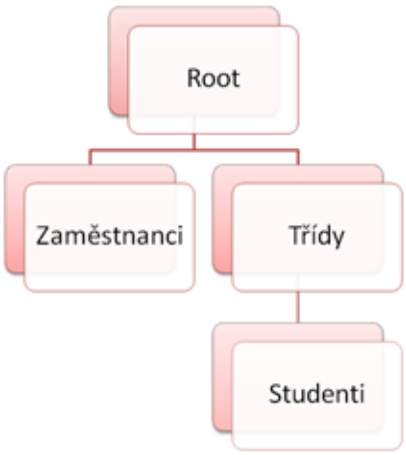
Autor (Tomáš Jecha 2009) rozdělil databáze podle struktury dat. Databáze se rozděluje na plochý databázový model, hierarchický databázový model, síťový databázový model, relační databázový model (též relační databáze) a objektový databázový model (též objektová databáze). Hierarchická a síťová databáze patří mezi první databáze a měly za úkol odstranění redundance dat.

3.2.1.1 Plochá databáze (prostý databázový soubor)

Plochá databáze je dle (Tomáš Jecha 2009) také označována jako prostý databázový soubor. Plochá databáze (Flat) patří mezi nejzákladnější typ, kde je sada záznamů. Databáze je určena pro uchovávání dat jako obyčejnou sadu záznamů. Specifikací ploché databáze je, že neobsahuje vazby, které vedou na jiné záznamy. Využití pro plochou je v jednoduchém ukládání seznamů. Příklad pro typ souboru, který patří pod plochou databázi, je csv soubor.

3.2.1.2 Hierarchický databázový model

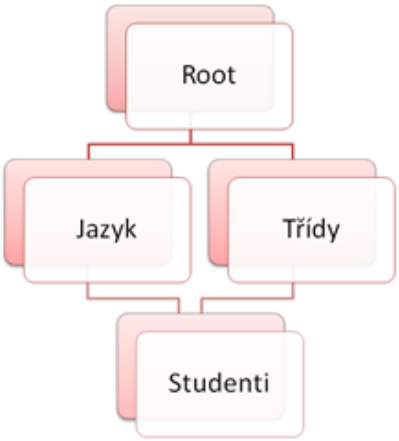
Hierarchický databázový model je společně se síťovou databází podle (netinbag 2022) předchůdce dnešních databází. Model byl vytvořen firmou IBM v 60. letech. Model se považuje za první databázový model, který byl vytvořen a nasazen. Modelu se říká one to many podle počtu uzlů, kde jeden uzel náleží mnoha uzlům, které jsou na něm závislé. Model lze využít v dnešní době pro databáze, kde se data neupravují a jsou určena na delší dobu, a pro jednoduché databáze s málo záznamy.



Obrázek 18-Hierarchická databáze, Zdroj: (cdn.dotnetportal. 2022)

3.2.1.3 Síťový databázový model

Internetový stránka (mariadb 2022) říká, že síťový model databáze vznikl o něco později než hierarchický model. Síťový model vychází z hierarchického datového modelu a upravuje některé jeho nedostatky. V síťové databázi je možné, aby podřízený prvek měl více nadřazených, což nešlo v předchozím modelu. Zde je tedy možné aplikovat složitější vztahy (many to many). Příkladem je databáze, která řeší vztah mezi objednávkami a díly. Nevýhoda v síťové databázi spočívá v obtížné implementaci. Další problém spočívá v požadavku na dobrou orientaci správce databáze ve struktuře dat.



Obrázek 19- Síťový databázový model, Zdroj: (cdn.dotnetporta 2021)

3.2.1.4 Relačně databázový model

Relační databázový model dle (Hernandez a John 2004) spočívá v definici vztahů. Vztahy jsou definovány pomocí tabulek. Relační databázový model patří v dnešní době mezi nejoblíbenější databázový model. Mezi nejznámější relačně databázové modely patří softwary MySQL, Oracle a Microsoft SQL Server.

3.2.1.5 Objektově databázový model

Autor (Peterson, guru99 2021) říká, že Objektový databázový systém umožnuje uložit všechny typy dat. Data se ukládají ve formě objektů. Uložené objekty v databázi obsahují atributy a metody, které určují, co má daný objekt dělat.

3.2.2 Relační databázový model

V praktické části bude relační databáze vytvořena pomocí jazyka SQL. Relační databáze bude vytvořena v programu PostgreSQL. Praktickou implementaci relačního databázového modelu obsahuje kapitola 4.3.2.

Vývoj relační databáze dle (Tomáš Jecha 2009) znamenal veliký průlom. Oproti hierarchickému a síťovému databázovému modelu, relační databázový model nenahlíží na databázový model z hlediska obsahu kořenového prvku v databázi, který se dále rozvětvuje. Relační databázový model má funkci pohlížet na všechny prvky (tabulky) na stejné úrovni. Základem pro relační databázový model je relace, což je výraz

pro databázové tabulky. Pro tento model je specifické ukládání dat do tabulek. Je zde nejčastěji použit jazyk SQL (Structured Query Language). Jazyku SQL je na trhu s mírnými úpravami mnoho. Relacím se někdy říká tabulky z důvodu, že relace se dá představit jako dvourozměrná datová struktura, která se skládá ze záhlaví, sloupců (atributů) a řádků, které představují konkrétní hodnoty (prvky). Pro webové aplikace se relační databáze používá hlavně MySQL v kombinaci s PHP.

3.2.2.1 Prvky relační databáze

Princip dle (Hernandez a John 2004) spočívá relační model v uložení dat ve vztazích (relacích). Pro chápání uživatele jsou tyto relace zobrazeny jako tabulky. Každý vztah mezi relacemi je zaznamenáván pomocí vektorů souřadnic (záznamů) a atributů (pole). Relační databázový systém také obsahuje charakteristiky klíče, pohledy a vztahy.

3.2.2.1.1 Relace (Tabulky)

Autoři (Hernandez a John 2004) představují relace. Relace (tabulky) představují základní strukturu relační databáze. Jednotlivé tabulky jsou jedinečné a představují specifický subjekt v databázi. V tabulce nezáleží na logickém pořadí záznamů a polí, protože všechny relace jsou na stejné úrovni. Tabulka je označována pomocí řádků a sloupců. Řádky představují pole záznamů. Sloupce představují atributy databáze. Každá tabulka musí obsahovat alespoň jeden sloupec, který je definován jako primární klíč. Účel tabulky je prezentování objektu anebo události. Objekt představuje hmatatelnou věc z reálného života (osoba, místo…). Každý objekt obsahuje ihned zpracovatelná data, která lze zpracovávat mnoha způsoby. Událost představuje určitou událost, ke které může za jistých okolností dojít. Událost má v sobě charakteristiky, které se poté zpracují podle přesně daných pravidel. Příklad pro událost je soudní slyšení, laboratorní testy.

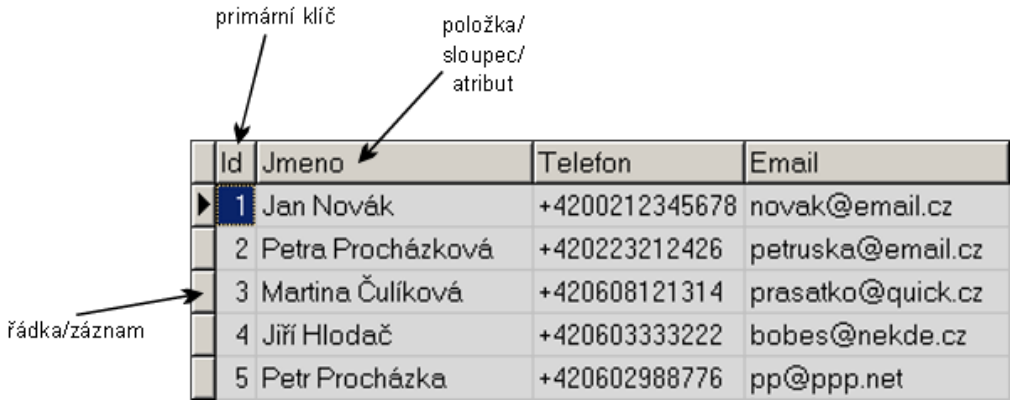
3.2.2.1.2 Atribut

Atribut (pole) dle (kme.vse 2008) zaznamenává sloupce v tabulce. Atribut slouží jako popsání vlastnosti záznamu. Každý atribut musí být označen jedinečným jménem a datovým typem. Datový typ se určí podle dat, která chce uživatel ukládat do relace (číslo, text, logická hodnota…). Všechny datové typy, které lze použít pro relační databázi, jsou

popsány dříve (kapitole 3.1.4.2). Pole vždy musí obsahovat pouze jednu hodnotu a definici typu hodnot, které mohou být zadány. Příklad pro atribut je, že třída Osoba bude mít atribut Bydliště, kde bude datový typ VARCHAR.

3.2.2.1.3 Záznamy

Autoři (Hernandez a John 2004) označují záznam jako jednoznačnou instanci subjektu tabulky. Hodnota v záznamu může být prázdná (NULL), kromě případů primárního klíče. Pro hodnotu NULL musí být u atributu zapsáno, že může být prázdný. Každý záznam má svůj primární klíč, který záznam jednoznačně odlišuje od ostatních. Záznam je důležitý z hlediska propojení více tabulek. Správce potřebuje vědět, jak záznam v jedné tabulce souvisí se záznamem v tabulce jiné.



Obrázek 20 Znázornění relace, Zdroj: (kosek 2022)

3.2.2.1.4 Klíče

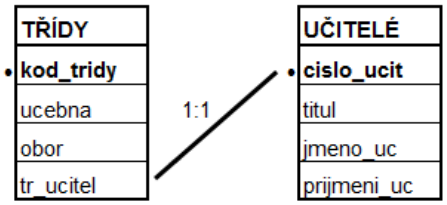
Dle internetové stránka (cs.education-wiki 2022) existuje v databázích SQL mnoho klíčů, které se dají použít. Klíče se řadí mezi speciální pole, které mají svoji specifickou roli. Nejdůležitější jsou dva klíče (primární a cizí). Mezi další klíče patří Unikátní klíč, Kandidátský klíč, Alternativní klíč a Složený klíč.

- Primární klíč (Primary key)

  Primární klíč dle autorů (Hernandez a John 2004) slouží jako jednoznačná identifikace každého záznamu v tabulce. Důležitost primárního klíče spočívá v jednoznačné identifikaci záznamů a identifikaci atributu v rámci celé databáze. Primárního klíč nemůže obsahovat hodnotu NULL. Obvykle je primární klíč značen zkratkou PK. Primární klíč musí být obsazen v každé tabulce (relaci).

- Cizí klíč (Foreigh key)

  Internetová stránka (cs.education-wiki 2022) popisuje cizí klíč k znázornění primárního tabulce v jiné relaci. Cizí klíč je používán z důvodu propojení dvou relací. Mezi primárním a cizím klíčem musí existovat vztah, že hodnoty z cizího klíče vycházejí z hodnot primárního. Cizí klíč může mít na rozdíl od primárního klíče duplicitní hodnoty a hodnotu NULL.

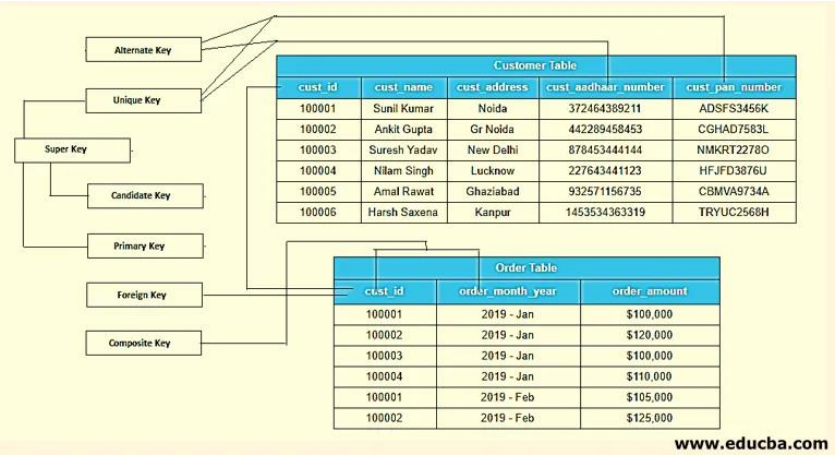


Obrázek 21- Znázornění propojení mezi primárním a cizím klíčem, Zdroj: (spsehavirov 2022)

- Kandidátský klíč

Kandidátský klíč představuje dle (difference-between-primary 2019) záznam (atribut), které funguje k jednoznačnému identifikování dat v relaci. O kandidátský klíč se jedná, když v jedné relaci mají všechny záznamy rozdílné hodnoty.

Internetová stránka (difference-between-primary 2019) popisuje rozdíl mezi primárním a kandidátským klíčem. Primární klíč může být pouze jeden a kandidátských klíčů může být více. Kandidátský klíč může mít hodnotu NULL. Primární klíč vychází z kandidátského klíče.



Obrázek 22-Kandidátský klíč, Zdroj: (cdn.education-wiki 2022)

- Složený klíč

  Složený klíč (zřetelný klíč) představuje podle zdroje (cs.education-wiki 2022) primární klíč, který je složen z více atributů. Složený klíč je určen k jednoznačné identifikaci relace. Výhoda spočívá v jednoznačné identifikaci pomocí více sloupců (atributů), kdy bez složení jeden sloupec by neznamenal jedinečnou identifikaci záznamu.

3.2.2.1.5 Pohledy (View)

Role pohledů autoři (Hernandez a John 2004) vysvětlují v SQL zobrazení virtuální tabulky. Uživatel má možnost si zobrazit tabulku, která je složená z atributů jedné nebo více relací v databázi. Vytvořený pohled je označován jako bázová (základní) relace. V relačním databázovém modelu se pohled bere virtuálně z důvodu převzetí dat z relací. Pohledy nejsou schopny žádná data samy uložit a databáze je uloží jako informaci. Výhodou pohledu je ve flexibilní možnosti práce s daty. Pomocí pohledu lze pochopit data z různých perspektiv. Pohledy se nejčastěji používají při požadavku zobrazení dat, která se nacházejí ve více relací.

3.2.2.1.6 Vztahy

Autoři (Hernandez a John 2004) vysvětlují 3 vztahy mezi dvěma relacemi. Je to jeden k jednomu, jeden k více a více k více.

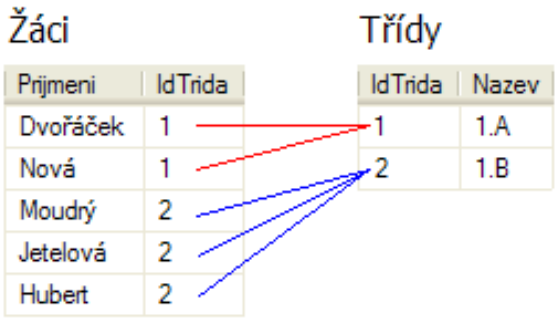
- 1:1

  Vztah jeden k jednomu popisuje (Hernandez a John 2004) jako vytvoření vztahu mezi dvěma relacemi v případě, když jeden záznam z první relace souvisí právě s jedním záznamem z druhé relace. V tomto druhu vztahu se označí jedna relace jako primární a druhá jako sekundární. Z primární tabulky je vybrán primární klíč, který je vložen jako cizí klíč do sekundární tabulky. Vztah jeden k jednomu není příliš častý. Nejčastější případ nastává při rozdělení jedné tabulky na dvě části z důvodu skrývání důvěryhodných informací.

- 1:více

  Autoři (Hernandez a John 2004) popisují případ vztahu 1:více. Pokud v jedné relaci je jeden záznam, který má v druhé relaci více záznamů, tak se používá vztah jeden k více. V druhé tabulce může být více záznamů, které mají souvislost pouze s jedním záznamem v první tabulce. Provedení spočívá v zaznamenání primárního klíče

ze vztahu jeden do relace, kde je více záznamů. V relaci s více záznamy se vytvoří cizí klíč z primárního klíče ze vztahu s jedním záznamem.



Obrázek 23-příklad SQL vztahu 1: více,Zdroj: (cdn.dotnetportal 2022)

- Více k více

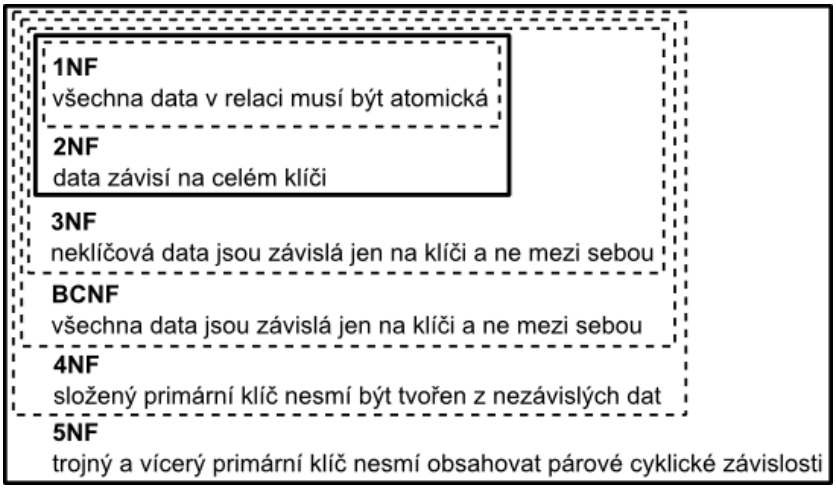
Vztah více:více autoři (Hernandez a John 2004) vysvětlují podle záznamů, kde záznamy z první relace patří k více záznamům druhé relace. Zároveň více záznamů z druhé relace patří k jednomu záznamu z první relace. Řešení problému více:více se řeší pomocí spojovací (vazební) tabulky. Princip spojovací tabulky spočívá v tom, že se vezmou primární klíče obou relací nebo vytvoření nového vlastního primárního klíče. Tyto primární klíče slouží jako složený klíč a tvoří strukturu nové tabulky. K vyřešení problému více:více je používána metoda spojování přes JOIN a tečkovou notaci v příkazu WHERE. Příkladem je databáze hudebních zpěváků a písní. Hudební zpěvák může mít více písní. A píseň může zazpívat více zpěváků.

3.2.2.2 Normalizace relační databáze

Podle (Qiu 2022) normalizace databáze slouží k jednoznačnému uspořádání dat v databázi. Do procesu normalizace patří vytvoření tabulek (relací) a vztahů mezi relacemi za předem definovaných pravidel, které fungují k ochraně dat. Další účel normalizace je odstranění redundance dat a nekonzistentní závislosti. Je dáno několik pravidel, které určují normalizaci databáze. Každé takové pravidlo má název normální forma.

Autoři (Kulhan a Lehocký 2008) představují normalizaci jako soubor pravidel, která by měla databáze splňovat. Každá splněná normální forma usnadňuje práci s daty (vybírání, aktualizace, manipulace…). Hierarchie normálních forem je uspořádána od nejnižší po nejvyšší. Každá vyšší forma obsahuje pravidla, která obsahují formy pod ní.

V některých případech platí, že dochází k tzv. splývání normálních forem, což znamená spojení některých normálních forem. Jako příklad pro splynutí normálních forem je použití jednoduchých klíčů (Obrázek 24-Normální formy v Relační databázi).



Obrázek 24-Normální formy v Relační databázi, Zdroj: (programujte 2022)

3.2.2.2.1 Normální formy

Normální formy fungují podle (geeksforgeeks 2022) dle hierarchie. Postupuje se od nulté normální formy až do páté. U všech normálních forem se předpokládá, že splňují podmínky předcházející a podmínky, které jsou v dané normální formě zadány. Existuje více než pět normálních forem, ale pro obecné popsání bude v kapitole normální formy popsáno pět normálních forem.

- Nultá normální forma (0NF)

  Dle (czwiki 2022) je nultá normální forma splněna skoro ve všech případech. V teorii se příliš neudává. Nultá normální forma udává, že v relaci je zadán alespoň jeden atribut, v kterém je zapsána alespoň jedna hodnota.

- První normální forma (1NF) - Jedinečnost polí

  První forma dle (Kalčev 2022) udává, že všechna data, která se nacházejí v relaci, musí být atomická (nedělitelná). Dalšími podmínky pro první normální formu je, že každá buňka (atribut) by měla obsahovat hodnotu, započítáno i NULL. V tabulce všechny pole musí představovat jedinečný typ informace.

- Druhá normální forma (2NF) - Primární klíče

  Autor (Kalčev 2022) popisuje, že všechny tabulky v databázi musí mít primární klíč (jedinečný identifikátor). Primární klíč může být vytvořen z jednoho či více atributů.

Data musí být závislá na celém klíči. Druhá normální forma určuje, že všechny záznamy (neklíčové atributy) musí být závislé na primárním klíči.

- Třetí normální forma (3NF) - Funkcionální závislost

Třetí normální forma dle autora (Kalčev 2022) udává, že v relaci není tranzitivní závislost. Třetí normální forma předpokládá, že všechna data v relaci jsou závislá pouze na primárním klíči. Všechna ostatní data slouží k popisu záznamu v tabulce.

- BCND (3,5NF)

(Kalčev 2022) udává formu BCND (Boyce-Coddova normální forma) jako dodatek ke třetí normální formě. V relaci nesmí být závislost primárního klíče. Podle BCND formy nesmí existovat mezi primárním klíčem a ostatními atributy závislost, kdyby se mohl primární klíč odvodit z ostatních dat.

- Čtvrtá normální forma (4NF) - Nezávislost polí

Čtvrtá normální forma udává nezávislost polí. V případě změny dat v libovolném poli, kromě primárního klíče, nesmí být ovlivněna ostatní data v relaci.

- Pátá normální forma (5NF)

Pátá normální forma určuje, že tabulka nelze více rozdělit bez ztráty dat. (czwiki 2022)

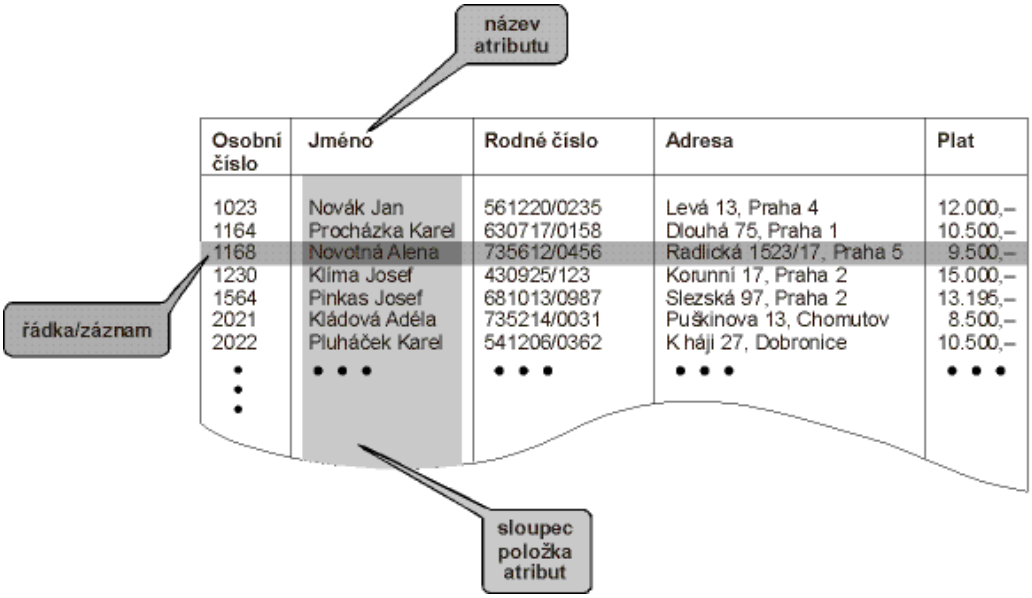
3.2.2.2.2 Redundance dat

Redundance dat je dle webu (netinbag 2022) stav, kdy jsou v databázi přítomná duplicitní data, čímž velikost databáze roste. Redundanci tvoří data, která jsou umístěna na více místech v databázi. Tyto duplicitní data není nutné mít v databázi z hlediska fungování databáze. Nevýhoda redundance dat je při práci s databází (načítání). Redundance dat může poškodit databázi. Přítomnost duplicitních dat může způsobit například ruční zadávání dat do databáze. Ochranu proti duplicitě už obsahuje mnoho databází a také existují softwarové programy pro jejich nalezení. V nejčastějším případě databáze přímo upozorní, že uživatel zadal duplicitní data.

Internetová stránka (netinbag 2022) udává příklad pro nevhodnost duplicitních dat. Databáze, která obsahuje informaci o zákaznících, obsahuje mnoho duplicitních dat zákazníků. V závislosti na databázi jsou tištěny poštovní štítky. Duplicita způsobí, že bude vytištěno mnoho štítků, které se budou opakovat.

- Vyřešení problému pole s více hodnotami

Problém nastává v případě podle autorů (Hernandez a John 2004), kdy jedna relace obsahuje atribut, kde se nachází více polí. Zjistit problém s více hodnotami je jednoduché. Oprávněný uživatel vezme každou relaci a zjistí, jestli nelze ji ještě nějakým způsobem rozdělit. Typickým případem je relace s atributem Jméno, kde jsou zapsány údaje jméno i příjmení. Více hodnot v jedné relaci může ohrozit integritu dat. Pokud se v databázi nacházejí relace, které mají atributy s více hodnotami, tak se problém vyřeší pomocí rozdělení atributu do té doby, než hodnota obsazená nepůjde dále zmenšit. Vyřešení Obrázek 25- Více hodnot v atributu Jméno spočívá v rozdělení atributu Jméno na dva atributy Jméno a Příjmení.



Obrázek 25- Více hodnot v atributu Jméno, Zdroj: (kosek 2022)

3.2.2.3 Práce s daty v relační databázi

3.2.2.3.1 Jazyk pro manipulaci s daty (Data Manipulation Language)

(techopedia 2014) definuje jazyk pro manipulaci s daty (DML). DML je určen jako sada příkazů, které umožní oprávněným uživatelům manipulovat s daty v databázi. Uživatel může vkládat, načítat, mazat a upravovat data. Jazyk pro manipulaci s daty se nejčastěji vyskytuje při práci s SQL databází. Výhoda DML je efektivnější interakce mezi uživatelem a systémem. Používané příkazy jsou SELECT, UPDATE, INSERT a DELETE.

- SELECT – Načtení dat z relační databáze. Nejpoužívanější příkaz DML.
- UPDATE – Upravení dat
- INSERT – Přidání záznamu do databáze
- DELETE – Odstranění záznamu z databáze

3.2.2.3.2 Jazyk pro definici dat (Data Definition Language)

Jazyk pro definici dat (DDL) je dle internetové stránky (techopedia 2020) používán jako sada příkazů k používání SQL databází. Účelem jazyku pro definici dat je vytváření a manipulace se strukturou databáze. Jazyk pro definici dat je určen pro vytvoření, změnu a odstranění objektů (indexů, pohledů, tabulek). Mezi nejpoužívanější příkazy patří CREATE, ALTER, TRUNCATE a DROP.

- CREATE – Příkaz pro vytvoření tabulky (relace).
- ALTER – Příkaz pro úpravu existující tabulky (relace). Příkaz slouží k přidání sloupce (odstranění), změnění datového typu.
- DROP – Příkaz pro odstranění tabulky (relace).
- TRUNCATE – Podobný příkazu DROP. Na rozdíl od DROPU zachová strukturu.
- RENAME – Přejmenování existujícího objektu v databázi
- COMMENT – Přidání komentáře do datového slovníku

3.2.2.3.3 Data pro kontrolu jazyka (Data Control Language)

(sql602.sourceforge 2022) definuje jazyk DCL pomocí příkazů GRANT a REVOKE. Příkazy slouží k úpravě práv a oprávnění. Práva se mohou přiřadit pro uživatele a skupinu uživatelů. Práva se přidělují k tabulce, speciálním záznamům nebo k proceduře. Uživatel může mít práva k mazání záznamů (DELETE), k vložení dat do záznamů (INSERT), ke čtení a k výpisu databáze (SELECT), k přepisování dat v záznamech (UPDATE), k poskytnutí dat jiným uživatelům (GRANT).

- GRANT – Příkaz k přidělení práv uživatelům k použití databáze
- REVOKE – Příkaz k odebrání přístupových práv (odebrání příkazu GRANT)

3.2.2.3.4 Jazyk ke kontrole transakcí (Transaction Control Language)

Příkazy jazyka se určují podle (geeksforgeek 2021) ke kontrole transakcí (TCL). Příkazy se zabývají transakcemi v databázi. Do jazyka se řadí příkazy COMMIT, ROLLBACK a SAVEPOINT.

- COMMIT – Potvrzuje transakci
- ROLLBACK – V případě chyby vrátí transakci zpět
- SAVEPOINT – V rámci transakce vytvoří bod uložení.

3.2.2.4 Příkazy v SQL

Příkazů pro SQL databáze je mnoho. Každý příkaz v databázi končí středníkem (;). Každý příkaz je značen velkými písmeny.

3.2.2.4.1 Příkaz SELECT

Příkaz SELECT je používán dle (javatpoint. 2022) v databázových systémech SQL jako příkaz, který se používá k vypisování a dotazu na data. SELECT je jedním z nejpoužívanějších příkazů, které se používají v databázových systémech. SELECT lze využít k přístupu k záznamům. Záznamy mohou být vybrány z jedné či více tabulek (relací) a pohledů. V příkazu SELECT se mohou také zadat podmínky pro jednotlivé atributy, podle kterých se má vyfiltrovat výsledná tabulka. Příkaz SELECT musí obsahovat vždy položku FROM. Pořadí v syntaxi je SELECT – FROM.

(javatpoint. 2022) popisuje nejjednodušší syntaxi příkazu SELECT jako: „SELECT * FROM Nazev_tabulky". Znak hvězdička znamená vypsání všech atributů, které se nacházejí v zobrazené tabulce. Pokud je žádoucí zobrazit pouze některé atributy, tak lze vyměnit znak hvězdičky za jména atributů. V příkazu SELECT lze přejmenovat názvy atributů, které se chce uživatel zobrazit. Tento způsob je vhodné použít, když jsou názvy atributů složité.

```
SELECT Název_sloupce_1, Název_sloupce_2, ....., Název_sloupce_N FROM Název_tabulky ;
```

Obrázek 26- Základní syntaxe příkazu SELECT, Zdroj: (javatpoint 2022)

- WHERE

Autoři (Hernandez a John 2004) popisují klauzuli WHERE v příkazu SELECT k filtraci dat. Zobrazené záznamy z databáze splňují podmínky, které jsou zapsány v příkazu WHERE. Příkaz WHERE se zapisuje v syntaxi za příkaz FROM. Příkaz WHERE se také používá při příkazech pro aktualizaci dat (UPDATE), změny dat v databázi (ALTER) a smazání dat a relací (DELETE). V položce WHERE je možno hodnoty porovnávat, zjistit rozsah mezi dvěma hodnotami, zjistit členství hodnoty v záznamu a shodu podle předem definovaných požadavků. Za WHERE lze přidat příkaz Limit, který určí, kolik se má vypsat výsledných záznamů.

- Logické operace v příkazu SELECT

  Podmínku WHERE dle (Hernandez a John 2004) může obsahovat jeden či více predikátů (podmínek). Po zadání predikátů vrací celý SELECT návratové hodnoty TRUE a FALSE. Vše dle splnění podmínek za podmínkou WHERE. Mezi jednotlivými predikáty se mohou zadat logické operátory AND a OR. Pro logický operátor AND (a zároveň) musí platit všechny podmínky. Pro logickou operaci OR musí platit alespoň jedna podmínka. Logické operátory lze použít i ve více operacích, než jsou dvě.

```
SELECT * FROM <tabulka1> WHERE <atribut1>='textovyretezec' AND <atribut2>=cislo;
```

Obrázek 27- Logické operace v příkazu SELECT, Zdroj: Vlastní zpracování

(Hernandez a John 2004) používají k vymezení rozsahu mezi hodnotami příkaz BETWEEN. Struktura BETWEEN je BETWEEN hodnota1 AND hodnota2 (´1986-07-01´ AND ´2005-06-02´). K nalezení dat, která mají splňovat podmínku podle definovaného řetězce se používá příkaz LIKE. Příkaz LIKE používá znaky %(procenta) a _(podtržítko). Podtržítko je představitel jednoho znaku. Procento určuje libovolný počet znaků včetně nuly. V příkazu SELECT lze vytvořit podmínku (IN), která je určena ke specifikaci, zda hodnota se vyskytuje v hledaných hodnotách.

Internetový zdroj (Shaw 2022) popisuje příkazy MAX, COUNT MIN, AVG a CONCAT v příkazu SELECT. Pro výpis maximální hodnoty číselného datového typu se používá funkce MAX, minimální hodnota je provedena pomocí MIN

a průměr pomocí AVG. Funkce CONCAT spojuje řetězce (například propojení jména a příjmení do jednoho řetězce).

Dle autorů (Hernandez a John 2004) hodnota NOT slouží k výpisu řádků, které znegují původní výrok. Hodnota NOT lze kombinovat s hodnotou NULL. Pomocí NOT NULL se vypíšou řádky, které nejsou prázdné.

- Příkazy COUNT, AVG a SUM

Internetového web (w3schools 2022) definuje použití funkce COUNT(), AVG() a SUM. SUM vrací sumu, která specifikuje počet řádků. Funkce COUNT vrací řádky dle předem definovaných specifik. Funkce AVG vrací hodnotu, která určuje průměr v číselném atributu. Funkce SUM slouží k vyjádření celkové sumy v celém atributu.

- Propojení tabulek (Relací)

Internetová stránka (techonthenet 2022) popisuje případy, kdy jsou potřeba z databáze data, která se nacházejí ve více tabulkách. K propojení více tabulek se používá příkaz JOIN a propojení pomocí příkazu FROM a WHERE. Příkaz JOIN lze rozdělit na INNER JOIN, LEFT JOIN, RIGHT JOIN a OUTER JOIN.

- INNER JOIN

  INNER JOIN se někdy nazývá jako jednoduché propojení. Jedná se o nejčastější příkaz propojení mezi více tabulkami. Vrácené záznamy jsou výsledkem průniku mezi tabulkami.

- LEFT JOIN

  Levý JOIN slouží k vrácení záznamů z vlevo zadané podmínky (před ON). Vrácené záznamy jsou všechny, které se vyskytují v levé relaci. Pokud je žádoucí mít pouze záznamy, které se vyskytují v levé relaci a nemají průnik v pravé relaci, dá se podmínka, že primární klíč z pravé relace je NULL.

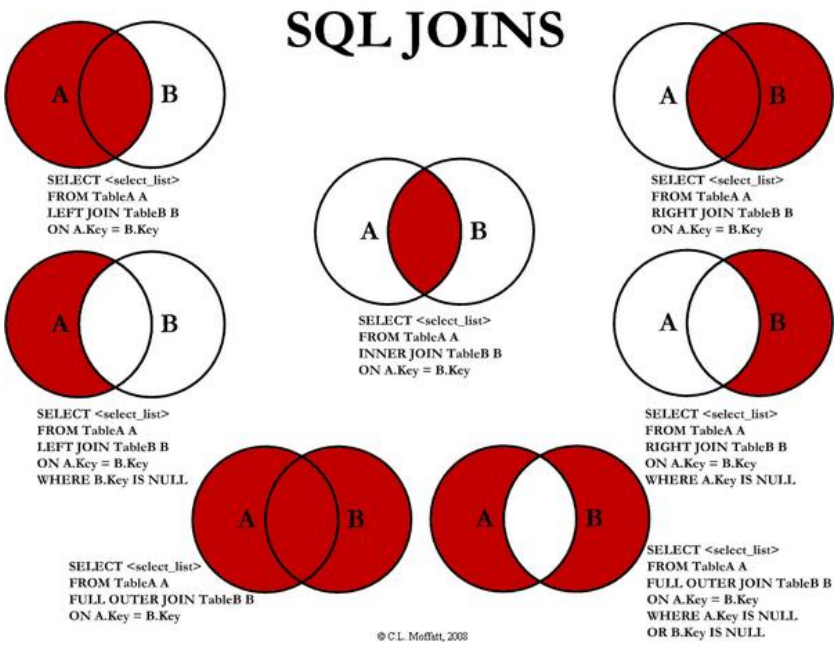
- RIGHT JOIN

  Pravý JOIN je stejný jako levý s rozdílem výběru dat z pravé relace. V tomto případě se vrací záznamy z tabulky, která je napsána za příkazem ON.

- OUTER JOIN

  OUTER JOIN funguje k vrácení všech záznamů mezi tabulkami. Lze zapsat pomocí příkazu WHERE, aby byly vráceny pouze rozdíly mezi jednotlivými

tabulkami. To se provede pomocí příkazu, který určí podmínku, že klíče jsou nulové (oba).



Obrázek 28- Propojení více tabulek v SQL pomocí JOIN, Zdroj: (i.stack.imgur 2022)

3.2.2.4.2 Příkaz UPDATE

Příkaz UPDATE v SQL je používán dle (w3schools 2022) pro úpravu záznamů, které se vyskytují v tabulkách. Příkaz UPDATE obsahuje podmínku WHERE, která definuje, jaké záznamy se mají aktualizovat. Pokud není zadaná podmínka (příkaz WHERE), jsou aktualizovány všechny záznamy v tabulce.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Obrázek 29- Syntaxe příkazu UPDATE, Zdroj: (w3schools 2022)

3.2.2.4.3 Příkaz INSERT

Příkaz INSERT v SQL databázi definuje (sql602.sourceforge 2022) jako příkazy, které vkládají data do tabulek. Při vkládání do tabulek je možné v závorce za jménem tabulky

napsat atributy, které mají být zapsány. Pokud nejsou v závorce za jménem třídy žádný atributy, tak se hodnoty vkládají v pořadí, jak jsou uvedeny atributy v relaci.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Obrázek 30- Syntaxe příkazu INSERT, Zdroj: (w3schools 2022)

3.2.2.4.4 Příkaz DELETE

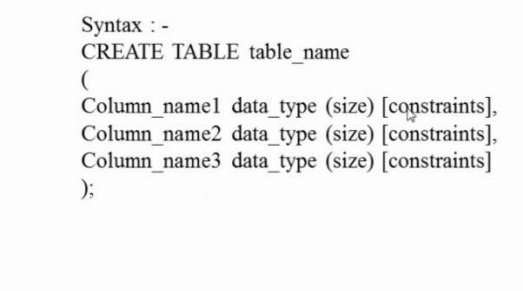
Příkaz DELETE je definován dle (w3schools. 2022) k odstranění záznamů z tabulek. Příkaz DELETE může obsahovat WHERE. Pokud není WHERE uveden, tak se vymažou všechny záznamy z tabulky splňující podmínku.

```
DELETE FROM table_name WHERE condition;
```

Obrázek 31- Syntaxe příkazu INSERT, zdroj: (w3schools 2022)

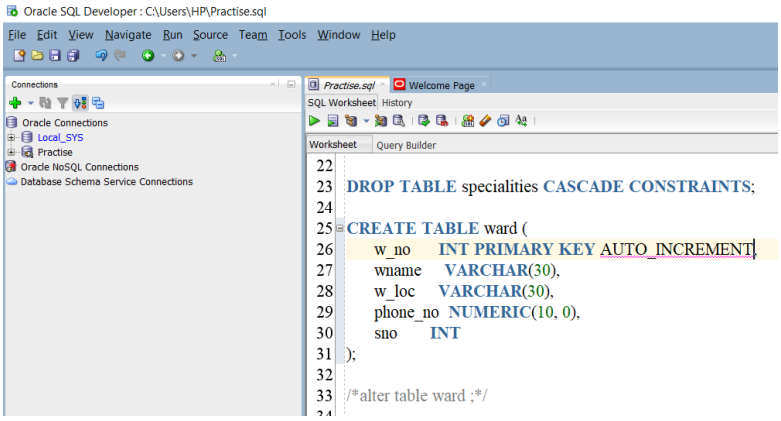
3.2.2.4.5 Příkaz CREATE

Článek od (Zedníček 2017) popisuje příkaz CREATE. Příkaz CREATE v SQL databázích slouží k vytvoření objektů. Relace, pohledy, databáze a pravidla jsou příklady, které mohou být vytvořeny pomocí příkazu CREATE. Pomocí příkazu se definují názvy relace, seznam sloupců a primární klíč. Primární klíč musí mít uživatel předem promyšlený. U seznamu sloupců se musí definovat datové typy. Každý sloupce ještě může být definován jako prázdný (NULL) nebo musí být vyplněný (NOT NULL). Mezi nejpoužívanější datové typy patří číslo (INT-INTEGER), řetězec (VARCHAR) a datum (DATE). Ke každému datovému typu je nutné přiřadit integritní omezení (jak velký počet míst může mít atribut). Do doplnění se vloží informace, zda je atribut primární klíč, jestli nemá být prázdný, anebo jestli nemá být automaticky vyplněn.



Obrázek 32-Syntaxe příkazu CREATE TABLE, Zdroj: (i.ytimg 2022)

Při vytváření relace lze za datový typ přidat i položku AUTO INCREMENT podle (w3schools. 2022). Při vkládání nového záznamu příkaz AUTO INCREMENT automaticky vygeneruje jedinečné číslo. Příkaz je používán nejčastěji v atributu, který je primárním klíčem.



Obrázek 33- Příklad vytvoření příkazu CREATE TABLE s položkou AUTO_INCERENT, Zdroj: (i.stack.imgur 2022)

Internetová stránka (w3schools. 2022) říká, že v příkazu CREATE lze přidat omezení CHECK. Omezení se používá k určení rozmezí hodnot, které mohou být umístěny v atributu.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

Obrázek 34- Přidání integritního omezení CHECK do CREATE TABLE, Zdroj: (w3schools 2022)

3.2.2.4.6 ALTER TABLE

Internetová stránka (w3schools. 2022) definuje příkaz ALTER TABLE k úpravě atributů v existující tabulce. Použití příkazu slouží k přidávání, odstranění a úpravě sloupců.

- Přidání atributu – Syntaxe: ALTER TABLE název_relace ADD název_sloupce datový typ.
- Smazání atributu – Syntaxe: ALTER TABLE název_relace DROP COLUMN název_sloupce.
- Úprava sloupce – K úpravě datového typu v existující tabulce. (w3schools. 2022)Syntaxe: ALTER TABLE název_relace MODIFY (COLUMN) název_sloupce datový typ.

3.2.2.4.7 DROP TABLE

Použití DROP TABLE používá (w3schools 2022) při odstranění existující relace v databázi. Syntaxe DROP TABLE je: „**DROP TABLE jméno_tabulky**;“.

3.2.2.4.8 TRUNCATE

Příkaz TRUNCATE definuje (w3schools 2022) jako podobný příkaz příkazu DELETE. Příkaz TRUNCATE slouží k vymazání dat v tabulce. Tabulka si zachová svoji strukturu. Syntaxe příkazu TRUNCATE je: **„TRUNCATE TABLE jméno_tabulky;**“.

3.2.2.4.9 RENAME

(javatpoint 2022) definuje příkaz k přejmenování existující tabulky v databázi. Syntaxe příkazu RENAME je: **„RENAME DATABASE staré_jmeno_databaze TO nové_jméno_databáze;**“.

3.2.2.4.10 Komentář

(w3schools. 2022) představuje komentáře při využívají vysvětlování SQL příkazů a k zamezení provedení příkazu. Komentář se může provést jako jednořádkový či více řádkový. Jednořádkový komentář se provede pomocí dvou pomlček (--). Víceřádkový komentář se musí provést na začátku a na konci. Komentář začíná dvojznakem /*. Komentář je zakončen dvojznakem */. Text mezi znaky bude ignorován.

3.2.3 Objektově databázový systém

Objektová databáze v praktické části práce je vytvořena pomocí programu Daskalos, který funguje na podobném principu jako SmallTalk. Diagram pro objektovou databázi vychází z kapitoly teoretické části (3.1.5). Ukázka praktického použití relační databáze je popsána v kapitole 4.4.2.

Hlavní rozdíl dle (thecustomizewindows 2021) mezi objektovou a relační databází je, že data v objektové databáze jsou založena na principu objektové orientace (objektů). Objektový databázový model je složen z objektové databáze a systému správy objektových databází (OODBMS).

Autor (Dancuk Milica 2022) pro zpracování dat (instance) v objektové databázi zpracovává instance jako kompletní celky (objekty). Pří práci s instancemi se pracuje v jednom velkém balíčku, kde je uloženo více objektů. Výhoda databáze spočívá v projektech, kde se pracuje se složitějšími daty. S objektovou databází se pracuje většinou společně s objektovým programováním.

Správu objektové databáze zpracovává autor (Dancuk Milica 2022) v objektově orientovaných systémech pro správu databází (OODBMS). Princip objektové databáze v sobě zachovává koncepty, které se využívají v objektově orientovaném programování a v principech relačních databází.

- Objekt

  Základní část autor (Merunka, Datové modelování 2022) definuje jako objekt. Objekt je modelován jako objekt reálného světa. Data, chování a vlastnosti jsou všechny uloženy v objektu. Popis objektu může být v rámci živé bytosti (člověk, zvíře), předmětu (osobní počítač, mobilní telefon, faktura) a abstraktním pojmu (diagnóza u lékaře, počasí).

  - Autor (Ondřej 2009) definuje vlastnosti objektů:
    - Objekt obsahuje své různé atributy, které je odlišují
    - Atributy mohou obsahovat přímé (hmotnost) či snadno změnitelné hodnoty (jméno)
    - Mnoho atributů má reprezentaci s jinými objekty. (dům – okna – rám, tabulka)
    - Rozdělení objektů lze dát do skupin stejného druhu (psi, domy …)

- V objektech funguje hierarchie (psi – psovité šelmy – šelmy…)
- Objekt dokáže aktivně reagovat na vnější vlivy

Objektová databáze je dle (Merunka, Datové modelování 2022) schopna použít více typů množin. Jako příklady množin lze vypsat Array, List, OrderedCollection, Set, Bag a Dictionary.

- Set

Kolekce Set znázorňuje množinu, kde prvky nejsou uspořádány. V případě přidání stejného prvku do databáze, zůstane v databázi vždy jednou. (Merunka, Datové modelování 2022)

- Bag

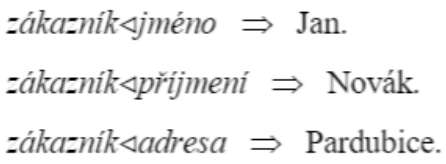
Ranec (Bag) slouží jako kolekce, kde nejsou prvky uspořádány. Rozdíl mezi kolekcí Set a Bag je, že při přidání stejného prvku, který se nachází již v databázi, bude v databázi více stejných prvků (kopií). (Merunka, Datové modelování 2022)

- List

Kolekce List je podobná kolekci Bag. V kolekci se ukládají kopie. V kolekci List lze nalézt uspořádání prvků, v jakém pořadí byly přidávány. (Merunka, Datové modelování 2022)

Objekt dle internetové stránky (Kumawat 2022) obsahuje strukturu, která představuje vlastnosti složení objektu. Název vlastností objektu jsou atributy. Tři komponenty jsou základní strukturou objektu.

- Zprávy – Vlastnost objektů autor (Merunka, Datové modelování 2022) definuje jako možnost reakce na požadavky, které jsou posílány. Tyto požadavky se nazývají zprávy. Zprávy, které jsou posílány, přijímají objekty, které se nazývají příjemce zpráv. V obrázku níže je uveden příklad pro zprávu, která se dotazuje objektu.



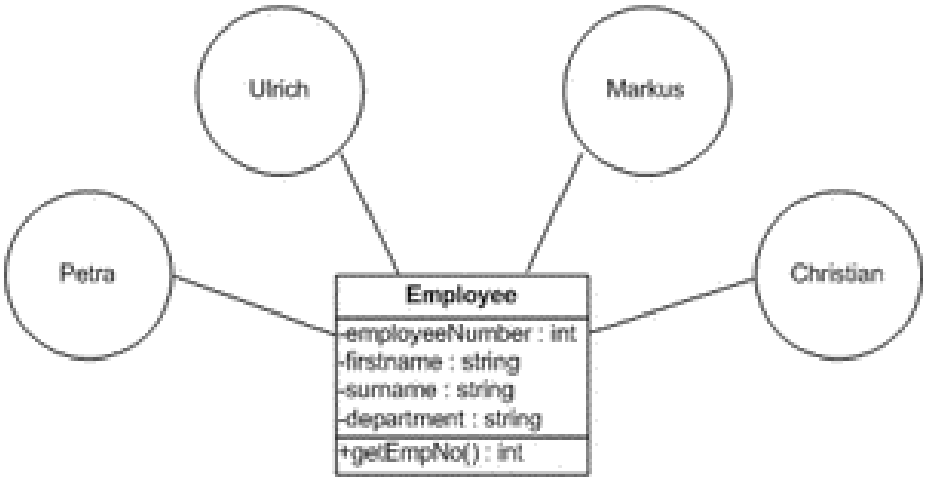
Obrázek 35- Zpráva na objekt v objektově orientované databázi, Zdroj: (docplayer 2022)

- Metody– (Merunka, Datové modelování 2022) říká, že metoda souvisí se zprávou. Předání zprávy způsobí, že se v těle kódu provedou instrukce, kterým se říká metody. Každé spuštění metody vyvolá hodnotu na výstupu.
- Data – Data představují uložené záznamy, které jsou v databázi. (Merunka, Datové modelování 2022)
- Proměnné – Proměnné ukládají data do objektu. Data, která jsou uložena v proměnných, jednoznačně odlišují jednotlivé objekty. – (Merunka, Datové modelování 2022)

- Třídy a Instance (typ)

V objektové databázi je nutné specifikování dle (Ondřej 2009) pojmu třída a instance. Pojem třída znamená sadu objektů, jejíž vnitřní struktura je přesně definována. Funkce třídy je k definování implementace objektů a typu pro jaký lze objekt použít. Důležitá vlastnost objektu je možnost změny třídy (atributů, operací) při zachování identity.

Internetový stránka (cs-exhibitions.uni-klu 2022) popisuje typ. Typ je potřeba k popsání sady objektů, které spolu sdílí stejné chování. Instance funguje na podobném principu jako je objekt. Instance nepatří pod třídu, ale je z třídy odvozena. Typ záleží na operacích, které lze vyvolat v objektu. Typem objektu mohou být jména, která vycházejí ze třídy Zaměstnanci.



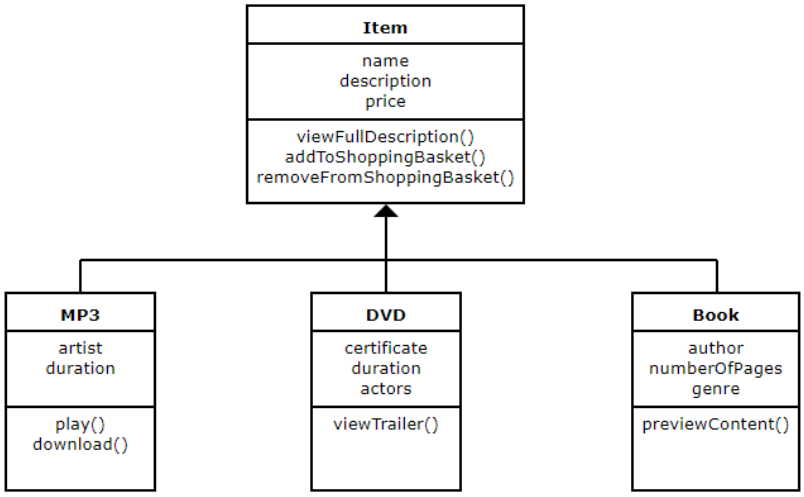
Obrázek 36- Třídy a objekty v objektové databázi, Zdroj: (cs-exhibitions 2022)

3.2.3.1 Koncepce objektových databází

(Dancuk Milica 2022) definuje pro objektově orientované databáze čtyři základní koncepce. Koncepce jsou dědění, polymorfismus, zapouzdření a abstrakce.

- Dědění

  Dědičnost je dle (Dancuk Milica 2022) vhodná v případě, kdy více objektů má stejné vlastnosti. V případě, kdy má více objektů více stejných vlastností a metod, je možno v objektové databázi vytvořit nadřazený objekt. Nadřazený objekt bude hierarchicky o úroveň výš. Nadřazený objekt obsahuje stejné vlastnosti, které oba (či více) objektů dědí a může využívat.



Obrázek 37- Dědění v objektové databázi, Zdroj: (101computing 2022)

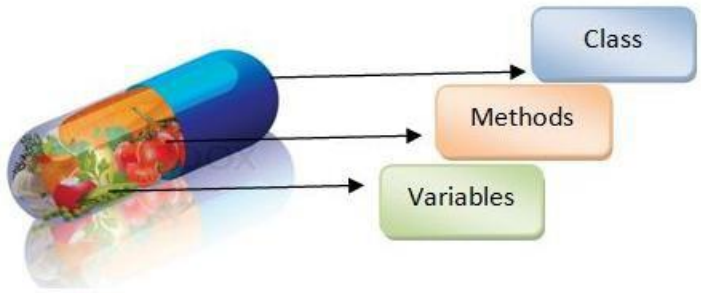
- Polymorfismus

Polymorfismus autor (Čápka, itnetwork 2022) zjednodušeně definoval jako schopnost objektu zaujímat více forem. Polymorfismus do jisté míry souvisí s děděním. Využití polymorfismu je v umožnění práce s různými typy objektů v jednotném rozhraní. Hlavní účel polymorfismu je metoda nadřazené třídy, která dědí stejnou hlavičku do podtříd, ale v podtřídách se změní tělo třídy dle požadavků. Příklad polymorfismu může být nadtřída zvíře, která obsahuje metodu mluvit. V každé podtřídě se dále metoda změní podle požadavků zvířete, jakými se dorozumívá.

Obrázek 38- Polymorfismus v objektové databázi, Zdroj: (itnetwork 2022)

- Zapouzdření

Autor (Dancuk Milica 2022) pro objektovou databázi definoval schopnost umožnovat princip zapouzdření. Princip zapouzdření je v seskupení dat a mechanismů do jednoho objektu. Zapouzdření v objektu způsobí ochranu přístupu k datům a mechanismům (objektům, metodám a ukazatelům). Pomocí zapouzdření uživatelé neznají, jak objekt funguje, a tím je lépe zabezpečený. Třídy jsou propojeny pomocí metod. Metody, které propojují objekty, není nutné sdílet a běžný uživatel nepotřebuje znát, jak fungují.



Obrázek 39- Zapouzdření v objektové databáze, Zdroj: (puntomarinero 2022)

- Abstrakce

Abstrakce v objektové databázi slouží pro zobrazení základní datové funkce. Zobrazí se datové funkce, které se zařadí do potřebné funkcionality. Jsou vybrány pouze informace, které jsou pro databázi důležité. Nedůležité informace jsou pro uživatele skryty. Abstrakce je určena pro snížení složitosti modelování dat a napomáhá pro opětovné použití. (Dancuk Milica 2022)

3.2.3.2 Operace s daty

Správná funkčnost objektové databáze se provede pomocí operací s kolekcemi objektů. V databázi se tyto operace jmenují dotazy nad bází dat. Pro operaci s daty se nejvíce používají selekce, kolekce a projekce.

- Selekce (Selection)

Selekce (výběr) funguje jako operace, pomocí které se z kolekce vybere podmnožina podle předdefinované podmínky. Formulace podmínky v selekci lze použít pomocí lambda-výrazu. (Merunka, Datové modelování 2022)

- Kolekce (Collection)

Kolekce (sběr dat) je určena k vybrání prvků, které se mají zobrazit v příkazu selekce. Kolekce vypíše výsledek, který je splněn pomocí selekce. (docs.mongodb 2022)

- Projekce

Operace méně používaná. Ve většině případů optimalizuje rozhraní mezi PC a uživatelem. Projekce funguje jako transformace prvků (zmenšení zobrazených dat). Projekce lze použít na omezení sloupců, které se mají zobrazit. Syntaxe pro kolekci je:

kolekce>> seznam atributů. V seznamu atributu se vypíše, které atributy se mají uživateli zobrazit. (Merunka, Datové modelování 2022)

3.2.3.3 Smalltalk

V diplomové práci je Smalltalk uveden z důvodu, že praktická práce v části objektové databáze bude provedena v programu Daskalos, který vychází a velice se blíží programu Smalltalk. (viz kapitola Objektová databáze)

Výhodou jazyka Smalltalk autor (Merunka, Datové modelování 2022) definuje v jednoduché syntaxi a dobré čitelnosti programu. Smalltalk je využíván v programování a práci s objekty. Dokáže zpracovat složení objektů, dědičnost, závislosti mezi objekty, polymorfismus. Smalltalk se řadí k nejlepším programovacím jazykům na bázi čistě objektově orientovaných. Smalltalk obsahuje pouze dva příkazy. Příkaz k pojmenování objektu a k posílání zprávy. K přiřazení hodnoty se používá operace pojmenování. Syntaxe je hodnota :=hodnota(přidělená).

3.2.3.3.1 Zprávy

Autor (Merunka, Datové modelování 2022) zprávy rozdělí na unární zprávy, binární zprávy a slovní zprávy. V případě chybějícího znaku kulaté závorky je nejvyšší priorita přiřazena unární zprávě.

- Unární zpráva
  - Syntaxe: objekt instance.
  - Unární zpráva je příkaz, který neposílá objektům žádná data.
- Binární zpráva
  - Dva objekty
  - Binární zpráva posílá příjemci jeden parametr.
  - První objekt (příjemce), druhý objekt (parametr zprávy)
  - Matematické operace (většinou vrací pravda a nepravda)
  - a◁plus: b.
- Slovní zpráva
  - Více parametrů než 1
  - K aplikování hodnoty a výpočtu metody na objekt

- Používají se při zápisu do databáze
- Lze i použít jako kaskádu zpráv v jednom řádku (kniha nazev: 'AAA'; rokvydani: 1850.
- Příklad: kniha název: 'AAA'.
  kniha rokvydani: 1850.

3.2.3.3.2 Selekce a kolekce

Selekce a kolekce se v Smalltalku podle (Merunka, Datové modelování 2022) používá v rámci bloků výrazů. Bloky jsou představeny v rámci lambda-výrazů. Blok výrazu představuje objekt, který lze pojmenovat a uživatel mu může poslat zprávy. V porovnání v syntaxi může být textový řetězec a číselný údaj (vše co se dá porovnat). V selekci může být i více podmínek najednou. Pro tyto účely se používají znaky & (a zároveň) a | (nebo). Příklad zapsání selekce:

3.2.3.3.3 Metody

Pro zápis metody se dle (Merunka, Datové modelování 2022) používá ve SmallTalku zápis, kde je hlavička metody, komentář k metodě v uvozovkách, pomocné objekty a tělo metody. Pomocné objekty obsahují objekty, s kterými se bude pracovat v metodě. Zpráva truncate v metodě určí k zobrazení celého čísla. Znak ^ (caret) určuje ve výrazu, že se z metody má vrátit hodnota. V metodách může být definováno, co se má provést v podmínce dle pravdivosti podmínky. Syntaxe takového příkazu je: logická hodnota ifTrue: [prikaz1] ifFalse: [prikaz2].

```
hlavička metody
"komentář metody"
        | seznam jmen pomocných objektů |
        tělo metody
```

Obrázek 40- Metoda ve Smalltalku, Zdroj: (Merunka, docplayer 26)

3.2.3.4 Daskalos

Daskalos je použit v praktické části pro objektové databáze. Daskolos autor (Merunka, Datové modelování 2022) popisuje jako fungující prezentaci objektově orientovaného modelování. Program je zdarma a vychází ze systému Smalltalk. V Daskalosu je možné

vytvářet třídy a metody. Objekty a třídy jsou znázorněny pomocí UML standartu (diagramu tříd).

```
(ZamestnanecSet select: [:x | (x vytvari vydavase prijmeni= 'Svetla') & (x vytvari vydavase
jmeno='Katerina') & (x vytvari datum= '01-06-2020' asDate)]) collect: [:x | x jmeno] with: [:x | x
prijmeni] with: [:x | x idzamestnance]
```

Obrázek 41- Příklad vypsání selekce a kolekce v objektové databázi Daskalos, Zdroj: Vlastní zpracování

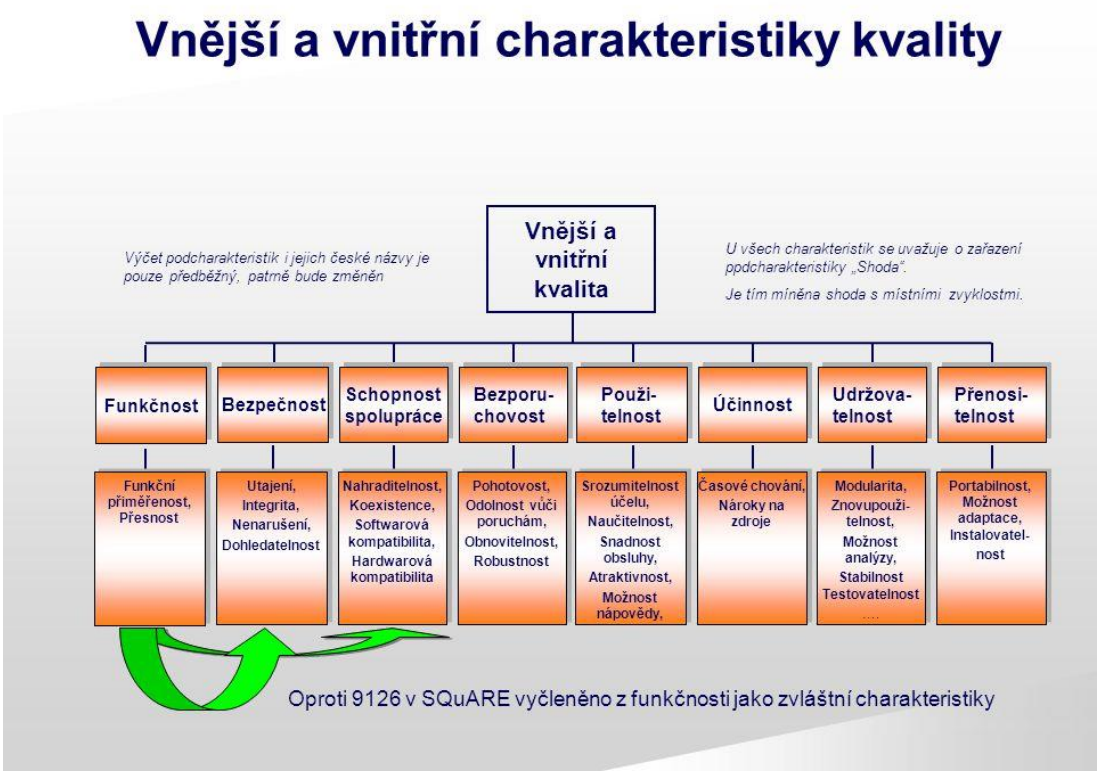
3.2.4 ISO/IEC 25000

Specializovaný systém pro celosvětovou normalizaci je tvořen Mezinárodní organizací pro normalizaci (ISO) a Mezinárodní elektronickou komisí (IEC). Na zpracování mezinárodních norem se podílí národní orgány, které jsou členy. (Systémové a softwerové inženýrství- Požadavky a hodnocení kvality systémů a softwaru (SQuaRE)- Pokyn ke SQuare 2017)

Účel normy (ČSN ISO/IEC 25000 2017) je poskytnutí hodnocení požadavků a hodnocení kvality systému a softwaru (SQuaRE). Název normy je Systémové a softwarové inženýrství – Požadavky a hodnocení kvality systémů a softwaru (SQuaRE). Cílovou skupinnou pro normu ISO/IEC 25000 jsou vývojoví pracovníci. (Systémové a softwerové inženýrství- Požadavky a hodnocení kvality systémů a softwaru (SQuaRE)- Pokyn ke SQuare 2017)

Norma je rozdělena v modelu Square podle čísel po desítkách. ISO/IEC 2500n slouží jako oddíl pro správu kvality. ISO/IEC 2501n je definován pro oddíly modelu kvality. ISO/IEC 2502n je definován pro oddíl měření kvality. ISO/IEC 2503n je definován jako oddíl požadavků na kvalitu, ISO/IEC 2504n oddíl pro hodnocení kvality a ISO/IEC 25050-25099 je oddíl, který popisuje rozšíření metody SQuaRE. (Systémové a softwerové inženýrství- Požadavky a hodnocení kvality systémů a softwaru (SQuaRE)- Pokyn ke SQuare 2017)

Obrázek 42-Charakteristiky kvality ISO/IEC 25000, Zdroj: znázorňuje parametry charakteristik pro ČSN ISO/IEC 25000. Z těchto parametrů byly vybrány charakteristiky, které byly použity v dotazníkovém šetření diplomové práce. (slideplayer 2022)



Obrázek 42-Charakteristiky kvality ISO/IEC 25000, Zdroj: (slideplayer 2022)

3.2.5 Dotazníkové šetření

Na internetové stránce (survio 2020) je vysvětlen návod a doporučení pro dotazníkové šetření. Dotazníkové šetření se provádí při semestrálních pracích, diplomových a bakalářských pracích, průzkumech trhu. Dotazník, který je špatně vytvořen, může mít negativní výsledky z hlediska mylných výsledků (negativních, pozitivních), sbíráním chybným dat.

Internetová stránka (survio 2020) doporučuje v dotaznících se vyhýbat citlivým otázkám, na které by respondent nemusel chtít odpovídat. Pokud je nutné mít v dotazníku citlivou otázku, doporučuje se například přidat rozmezí – například v otázce platu je to vhodnější než přesná částka. Doporučení také spočívá v neomezování respondentů (například v možnosti odpovědět pomocí dvou uzavřených odpovědí). V dotazníku by měla být otázka vždy položena kladnou formou. Používání odpovědí pomocí neurčitých slov (často, někdy, nikdy) je složité z důvodu, že každý člověk má tyto slova jinak

definována. Otázky v dotazníku lze provést pomocí otevřených, polootevřených a uzavřených otázek.

4 Vlastní práce

Praktická část se týká tvorby relační a objektové databáze pro teoretický klub HC Lvi Příbram a jejich porovnání. V první analytické části jsou shrnuta požadovaná data a výstupy. Druhá část obsahuje návrh Diagramu tříd, který slouží jako předpoklad pro objektovou a relační databázi. Další část obsahuje vytvořené implementace v objektové a relační databázi. U obou částí jsou vysvětleny postupy pro práci s danými modely. Relační model je podpořen Entity Relationship diagramem z důvodu znázornění vazebních tabulek. Diagram tříd a implementační E-R diagram byl vytvořen v programu StarUML. Po návrhu objektové a relační databáze je vytvořen dotazník pro porovnání. Dotazník je složen z dotazů dle vybraných parametrů dle ČSN ISO/IEC 25000.

Všechna data v hokejovém týmu jsou smyšlená a neobsahují žádná reálná data. Z důvodu použití v diplomové práci, která je dostupná jako veřejný dokument, jsou všechna data vygenerována. V databázi jsou uložena všechna data, ale pro výpis dat z klubu (viz níže), jsou vypsána jen data, která neobsahují důvěryhodné informace. Všechny informace osob byla vygenerována pomocí internetové stránky http://www.jmenaprijmeni.cz/generator-jmen-online. V reálné databázi musí být všechna osobní data kódována dle principů GDPR. Rodné číslo a datum narození byly generovány z internetové stránky https://webdev.zaujimave.info/generator-rodneho-cisla/.

Relační databáze je znázorněna pomocí implementačního E-R diagramu. Diagram neobsahuje vazbu mezi tabulky M:N. Vztah M:N je znázorněn pomocí vazební tabulky. Vazební tabulka znázorňuje reálné zpracování relačního datového modelu.

Objektová databáze je znázorněna pomocí diagramu tříd. Diagram tříd pro objektovou databázi je vygenerován pomocí objektového programu Daskalos.

4.1 Klub HC Lvi Příbram

Pro účely diplomové práce byl vytvořen fiktivní hokejový klub s názvem HC Lvi Příbram. Další data pro testování byla zvolena. Jsou známy základní informace o klubu (Tabulka 5-Informace o klubu HC Lvi Příbram, Zdroj: Vlastní zpracování).

HC Lvi Příbram			
Rok založení	*Název stadionu*	*Město stadionu*	*Adresa stadionu*
1895	Ovál	Příbram	U roury 1530

Tabulka 5- Informace o klubu HC Lvi Příbram, Zdroj: Vlastní zpracování

- Představenstvo klubu

Činnost klubu řídí představenstvo. Jsou navoleny funkce a přiřazeny osoby s údaji o telefonním čísle a e-mailu z důvodu kontaktování veřejnosti (Tabulka 6- Představenstvo klubu HC Lvi Příbram, Zdroj: Vlastní zpracování).

Představenstvo klubu				
Jméno	**Příjmení**	**Funkce**	**Telefonní číslo**	**Email**
Jonáš	Černík	Prezident klubu	414 514 321	
Soběslav	Kožíšek	Sportovní manažer	448 888 888	sobeslav.kozisek@lvipribram.cz
Marek	Zavadil	Vedoucí mládeže	111 888 687	zavadil@lvipribram.cz
Jana	Svátková	Marketingová ředitelka	547 880 000	j.svatkova@lvipribram.cz
Kateřina	Kolová	Finanční ředitelka	555 555 555	k.kolova@lvipribram.cz

Tabulka 6- Představenstvo klubu HC Lvi Příbram, Zdroj: Vlastní zpracování

- Zaměstnanci

Pro klub pracují zaměstnanci. Ti se starají o jednotlivé kategorie klubu. Každý zaměstnanec může pracovat ve více kategoriích. Pro klub pracují také zaměstnanci, kteří pracují jako rolbař a správce stadionu (Tabulka 7- Zaměstnanci klubu HC Lvi Příbram, Zdroj: Vlastní zpracování).

Zaměstnanci klubu			
Jméno	**Příjmení**	**Funkce**	**Funkce1**
Čeněk	Masař	Hlavní trenér A-tým	Asistent trenéra Junioři
Prokop	Vaculík	Hlavní trenér 1+2 třídy	Vedoucí mužstva Starší žáci
Michal	Tomáš	Hlavní trenér 3+4 třídy	
Robin	Šír	Hlavní trenér Náboru	Hlavní trenér 5 třídy
Štěpán	Smolka	Hl. trenér Juniorů	Asistent trenéra Dorostu
Oldřich	Šefčík	Hlavní trenér Starších žáků	Správce stadionu
Oto	Vorlíček	Vedoucí mužstva A-týmu	Asistent trenéra Mladších žáků
Milan	Takáč	Rolbař	
Eliška	Němcová	Vedoucí mužstva Dorost	
Natálie	Králová	Vedoucí mužstva Junioři	
Tomáš	Pecha	Asistent trenéra A-tým	Hlavní trenér Dorostu
Marcel	Turek	Hlavní trenér Mladší žáci	

Tabulka 7- Zaměstnanci klubu HC Lvi Příbram, Zdroj: Vlastní zpracování

- Hráči

Hráči jsou do kategorií rozděleny dle věku. Věkové kategorie v týmu HC Lvi Příbram jsou Nábor, 1. + 2. třída, 3. + 4. třída, 5. třída, Mladší žáci, Starší žáci, Dorost, Junioři a A-tým. Ke každému hráči jsou zadané údaje jméno, příjmení, váha, výška, rodné číslo, datum narození, konec platnosti registrační karty

Hráči klubu HC Lvi Příbram							
Jméno	**Příjmení**	**Kategorie**	**Konec reg.kar.**	**Jméno**	**Příjmení**	**Kategorie**	**Konec reg.kar.**
Novak	Novák	Nábor		Roman	Šlajs	Starší žáci	S450123
Karel	Němec	Nábor		Hanuš	Bubeník	Starší žáci	S450124
Jan	Trouba	Nábor		Štěpán	Nývlt	Starší žáci	S450125
Dobroslav	Hermann	Nábor		Květoslav	Vykoukal	Dorost	S450126
Slavomír	Nový	Nábor		Emil	Holomek	Dorost	S450127
Mojmír	Šebera	Nábor		Vilém	Kotala	Dorost	S450128
Cyril	Hnátek	Nábor		Mojmír	Zíma	Dorost	S450129
Přemysl	Vajda	Nábor		Lubomír	Vrabec	Dorost	S450130
Jan	Kropáč	Nábor		Emil	Jankovský	Dorost	S450131
Jonáš	Brzák	1+2 třída	S450100	Leoš	Vorel	Dorost	S450132
Vlastimil	Mlčák	1+2 třída	S450101	Mojmír	Borovička	Dorost	S450133
Kryštof	Vyroubal	1+2 třída	S450102	Jiří	Ducháček	Junioři	S450134
Daniel	Kopeček	1+2 třída	S450103	Martin	Ruml	Junioři, A-tým	S450135
Zdeněk	Tesař	1+2 třída	S450104	Luboš	Vágner	Junioři	S450136
Erik	Janečka	3+4 třída	S450105	Hynek	Drbal	Junioři	S450137
Ladislav	Šedivý	3+4 třída	S450106	Kryštof	Valášek	Junioři	S450138
Vít	Kulík	3+4 třída	S450107	Jan	Ondra	Junioři	S450139
Rostislav	Šmerda	3+4 třída	S450108	Luděk	Zelený	Junioři	S450140
Ivan	Čapek	3+4 třída	S450109	Šimon	Král	A-tým	S450141
Tomáš	Měrka	3+4 třída	S450110	Ivan	Knotek	A-tým	S450142
Jáchym	Hůlka	5 třída	S450111	Mikuláš	Vráblík	A-tým	S450143
Iveta	Martincová	5 třída	S450112	Karel	Němec	A-tým	S450144
Jana	Sedláčková	5 třída	S450113	Stanislav	Pokorný	A-tým	S450145
Martina	Blažková	5 třída	S450114	Michal	Tomáš	A-tým	S450146
Karolína	Smrková	Mladší žáci	S450115	Tomáš	Němý	A-tým	S450147
Dalimij	Blažek	Mladší žáci	S450116	Robert	Krysa	A-tým	S450148
Roman	Zach	Mladší žáci	S450117	Boris	Stehno	A-tým	S450149
Stanislav	Macko	Ml. a St. Žáči	S450118	Aleš	Klobása	A-tým	S450150
Emil	Kosík	Mladší žáci	S450119	Miroslav	Červený	A-tým	S450151
Květoslav	Tobola	Starší žáci	S450120	Miloslav	Lupen	A-tým	S450152
Vlastimil	Pliska	Starší žáci	S450121	Dominik	Kreslený	A-tým	S450153
Ignác	Cejnar	Starší žáci	S450122				

Tabulka 8- Výpis hráčů podle kategorie, Zdroj: Vlastní zpracování

Položka konec registrační karty je zadána od 1. + 2. třídy dle požadavku, protože hráči v kategorii Nábor nemusí mít registrační kartu. Je možné zařazení hráčů do více kategorií. V databázi hokejového týmu ve více týmech hrají hráči Martin Ruml a Stanislav Macko.

- Kluby v databázi

Pro potřeby zápasů jsou v databázi uložena data o hokejových týmech, se kterými hrají zápasy HC Lvi Příbram (Tabulka 9- Týmy uložené v databázi, Zdroj: Vlastní zpracování). Je zadaný celý název klubu, rok založení, název města a stadion, kde hrají své domácí zápasy. Na jednom stadionu je možné, aby hrálo více týmů.

Soupeři HC Lvi Příbram			
Název klubu	**Název zimního stadionu**	**Město stadionu**	**Adresa stadionu**
HK Písek	U krbu	Písek	Slepá 412
HC Mirotice	Zimní stadion Blatná	Blatná	U kočky 701
HC Žraloci Blatná	Zimní stadion Blatná	Blatná	U kočky 701
HK Bludov	Bludovský stadion	Bludov	Jarní 20
HC Dobřichovice	Rajec aréna	Praha	Na střeše 325
HC Obři Český Rudolec	Zimní stadion v Telči	Telč	Krbní 19
HK Kostelec nad Vltavou	Kostelecký ležák	Kostelec nad Vltavou	Pivovarská 894
HK Čerti Hlohovice	Ovál	Příbram	U roury 1530
HC Zaječov	Tvrz	Zaječov	K sportovišti 11

Tabulka 9- Týmy uložené v databázi, Zdroj: Vlastní zpracování

- Rozpis zápasů pro kategorie

Zápasy pro jednotlivé kategorie se řídí podle rozpisu. Rozpis je generován automaticky a je rozlosován ve stylu, kdy jeden tým hraje jednou doma a jednou venku. Rozpis je vždy zadán před aktivní sezónou. Pro ukázku diplomové práce jsou rozepsány první 3 kola každé kategorie. U každého zápasu je zadán datum zápasu, soupeř a stadion (Tabulka 10- Rozpis zápasů, Zdroj: Vlastní zpracování).

Rozpis zápasů podle kategorie				
Kategorie		***1 kolo***	***2 kolo***	***3 kolo***
Nábor	*Soupeř*	HK Písek	HC Žraloci Blatná	HK Kostelec nad Vl.
	Stadion	Ovál	Zimní stadion Blatná	Ovál
	Domácí/Hosté	Domácí	Hosté	Domácí
1+2 třída	*Soupeř*	HC Mirotice	HC Obři Český Rudolec	HK Kostelec nad Vl.
	Stadion	Ovál	Zimní stadion v Telči	Ovál
	Domácí/Hosté	Domácí	Domácí	Domácí
3+4 třída	*Soupeř*	HC Mirotice	HC Dobřichovice	HC Obři Český Rudolec
	Stadion	Bludovský stadion	Rajec aréna	Ovál
	Domácí/Hosté	Hosté	Hosté	Domácí
5 třída	*Soupeř*	HK Bludov	HK Čerti Hlohovice	HC Zaječov
	Stadion	Bludovský stadion	Ovál	Ovál
	Domácí/Hosté	Hosté	Hosté	Domácí
Mladší žáci	*Soupeř*	Český Rudolec	HK Písek	HC Dobřichovice
	Stadion	Zimní stadion Blatná	U krbu	Ovál
	Domácí/Hosté	Hosté	Hosté	Domácí
Starší žáci	*Soupeř*	HK Kostelec nad Vl.	HC Obři Český Rudolec	HC Žraloci Blatná
	Stadion	Kostelecký ležák	Zimní stadion v Telči	Ovál
	Domácí/Hosté	Hosté	Hosté	Domácí
Dorost	*Soupeř*	HC Mirotice	HK Bludov	HC Obři Český Rudolec
	Stadion	Ovál	Bludovský stadion	Ovál
	Domácí/Hosté	Domácí	Hosté	Domácí
Junioři	*Soupeř*	HC Zaječov	HK Čerti Hlohovice	HK Písek
	Stadion	Tvrz	Ovál	U krbu
	Domácí/Hosté	Hosté	Hosté	Hosté
A-tým	*Soupeř*	HC Zaječov	HK Kostelec nad Vl.	HC Dobřichovice
	Stadion	Tvrz	Ovál	Rajec aréna
	Domácí/Hosté	Hosté	Domácí	Hosté

Tabulka 10- Rozpis zápasů, Zdroj: Vlastní zpracování

Při znalosti rozlosování soupeřů je nutné mít v databázovém systému i data s konkrétními informacemi o zápase jako jsou rozhodčí, lékař, datum a čas zápasu (Obrázek 43-Detailní informace o zápasech, Zdroj: Vlastní zpracování).

Informace o zápase						
Kategorie	**Kolo**	**Hlavní rozhodčí**	**Čárový rozhodčí**	**Lékař**	**Datum zápasu**	**Čas zápasu**
Nábor	*1 kolo*	Hříbal		Kratochvílová	25.09.2021	8:00
	2 kolo	Rejzek		Vrba	05.10.2021	8:00
	3 kolo	Škvor		Kratochvílová	12.10.2021	10:00
1+2 třída	*1 kolo*	Hříbal		Kratochvílová	05.10.2021	10:00
	2 kolo	Hříbal		Kratochvílová	05.10.2021	12:00
	3 kolo	Hříbal		Kratochvílová	05.10.2021	14:00
3+4 třída	*1 kolo*	Babka		Kohut	21.09.2021	13:00
	2 kolo	Kohut		Pospíšil	28.09.2021	10:00
	3 kolo	Hříbal		Stříž	28.10.2021	9:15
5 třída	*1 kolo*	Dufek	Páral, Škvor	Hamáček	20.09.2020	10:00
	2 kolo	Chyba	Valeš,Zdeněk	Pospíšil	24.09.2020	14:00
	3 kolo	Nečas	Chyba, Žaloudek	Krestová	27.09.2020	15:00
Mladší žáci	*1 kolo*	Vrána	Vrána, Nešport	Herzán	01.09.2020	12:00
	2 kolo	Nečas	Chyba, Louda	Krestová	07.09.2020	15:00
	3 kolo	Nečas	Louda, Žaloudek	Krestová	14.09.2020	12:45
Starší žáci	*1 kolo*	Mrázek	Varvruška, Mrázek	Břicháčková	11.09.2020	13:00
	2 kolo	Jirouš	Lekeš, Šiman	Kročil	18.09.2020	13:00
	3 kolo	Chyba	Vlášek, Jirouš	Kerstová	25.09.2020	14:15
Dorost	*1 kolo*	Zavadil	Nešport, Drápal	Krestová	01.10.2020	18:15
	2 kolo	Kopřiva	Punčochář, Valeš	Louda	07.10.2020	17:45
	3 kolo	Šimčík	Ledvinka, Příhoda	Krestová	14.10.2020	15:30
Junioři	*1 kolo*	Klimeš	Komárek, Forman	Barton	01.10.2020	17:30
	2 kolo	Stoklásek	Rozehnal, Reichl	Princ	08.10.2020	17:45
	3 kolo	Štěch	Větrovský, Zdeněk	Volejník	15.10.2020	16:30
A-tým	*1 kolo*	Jirka	Klement, Janota	Čížek	02.10.2020	19:00
	2 kolo	Rýdl	Penáz, Lupáčková	Lang	16.10.2020	17:00
	3 kolo	Vávra	Švéda, Tupý	Voborník	21.10.2020	18:45

Obrázek 43-Detailní informace o zápasech, Zdroj: Vlastní zpracování

- Zápas

Jako ukázka zadání zápasu je zadané 1. kolo A-týmu. Soupeřem A-týmu je tým HC Zaječov. Utkání se odehrálo na stadioně Tvrz v Zaječově, 2.10. 2020 v 19:00. Utkání skončilo výsledek 6:3 pro tým HC Lvi Příbram. Na zápase byl přítomný hlavní trenér Čeněk Masař, asistent trenéra Tomáš Pecha a vedoucí mužstva Oto Vorlíček.

Ke každému zápasu je možné vybrat sestavu z dané kategorie a přiřadit hráčské posty. Také je možné zvolit kapitána a jeho dva asistenty.

Na ukázkový zápas byla vybrána sestava, která má 12 hráčů. V soupisce se nacházejí 2 brankáři, 4 obránci a 6 útočníků. Kapitánem týmu je útočník Ivan Knotek a jeho asistenti obránci Michal Tomáš a Boris Stehno (Tabulka 11- Soupiska A-tým na zápas proti HC Zaječov, Zdroj: Vlastní zpracování).

A tým vs Zaječov				
Jméno	**Příjmení**	**Číslo dresu**	**Pozice hráče**	**Funkce hráče**
Martin	Ruml	15	Útočník	
Šimon	Král	20	Útočník	
Ivan	Knotek	10	Útočník	Kapitán
Mikuláš	Vráblík	52	Útočník	
Karel	Němec	12	Útočník	
Stanislav	Pokorný	2	Útočník	
Michal	Tomáš	4	Obránce	Asistent kapitána
Tomáš	Němý	3	Obránce	
Robert	Krysa	13	Obránce	
Boris	Stehno	41	Obránce	Asistent kapitána
Aleš	Klobása	1	Brankář	
Miroslav	Červený	39	Brankář	

Tabulka 11- Soupiska A-tým na zápas proti HC Zaječov, Zdroj: Vlastní zpracování

Ze zápasu jsou zaznamenány statistiky branek, asistencí a trestů pro jednotlivé hráče. U trestů je evidovaný typ trestu, čas a jeho délka. V zápase měl nejvíce bodů Boris Stehno s 1 brankou a dvěma asistencemi (Tabulka 12- Statistiky zápasu A-tým vs Zaječov, Zdroj: Vlastní zpracování).

Branky v zápase			
Třetina	**Čas**	**Střelec**	**Asistent**
1	4:23	Němec	Ruml, Vráblík
1	19:31	Ruml	
2	9:45	Vráblík	Stehno, Král
2	11:10	Ruml	Král, Stehno
2	17:05	Stehno	
3	5:01	Tomáš	Pokorný, Němý

Tabulka 12- Statistiky zápasu A-tým vs Zaječov, Zdroj: Vlastní zpracování

V zápase jsou zaznamenávány trestné minuty. Nejvyšší trest zaznamenal hráč Karel Němec s 20minutovým trestem za nesportovní chování (Tabulka 13- Trestné minuty ze zápasu A-týmu, Zdroj: Vlastní zpracování).

Trestné minuty					
Hráč	**Číslo dresu**	**Název trestu**	**Délka trestu**	**Třetina**	**Minuta**
Martin Ruml	15	Podražení	2	2	15:10
Martin Ruml	15	Nedovolené bránení	2	3	17:10
Šimon Král	20	Napadení	2+2	3	18:40
Karel Němec	12	Nesportovní chování	10	3	18:40

Tabulka 13- Trestné minuty ze zápasu A-týmu, Zdroj: Vlastní zpracování

Ze zápasu jsou také zapisovány statistiky brankářů, kde oba brankáři dosáhli procentuální úspěšnosti zákroku kolem 90% (Tabulka 14- Brankářské statistiky ze zápasu A-týmu, Zdroj: Vlastní zpracování).

Statistiky brankářů				
Brankáři	**Číslo dresu**	**Počet střel**	**Obdržené góly**	**Průměr**
Miroslav Červený	39	20	2	90,00 %
Aleš Klobása	1	14	1	92,86 %

Tabulka 14- Brankářské statistiky ze zápasu A-týmu, Zdroj: Vlastní zpracování

Pro zápasy jsou naplněna data s tresty a jejich délkou, které může hráč mít ve svém vyloučení. U trestu jsou data s názvem trestu, délkou a typem (Tabulka 15- Tresty, Zdroj: Vlastní zpracování).

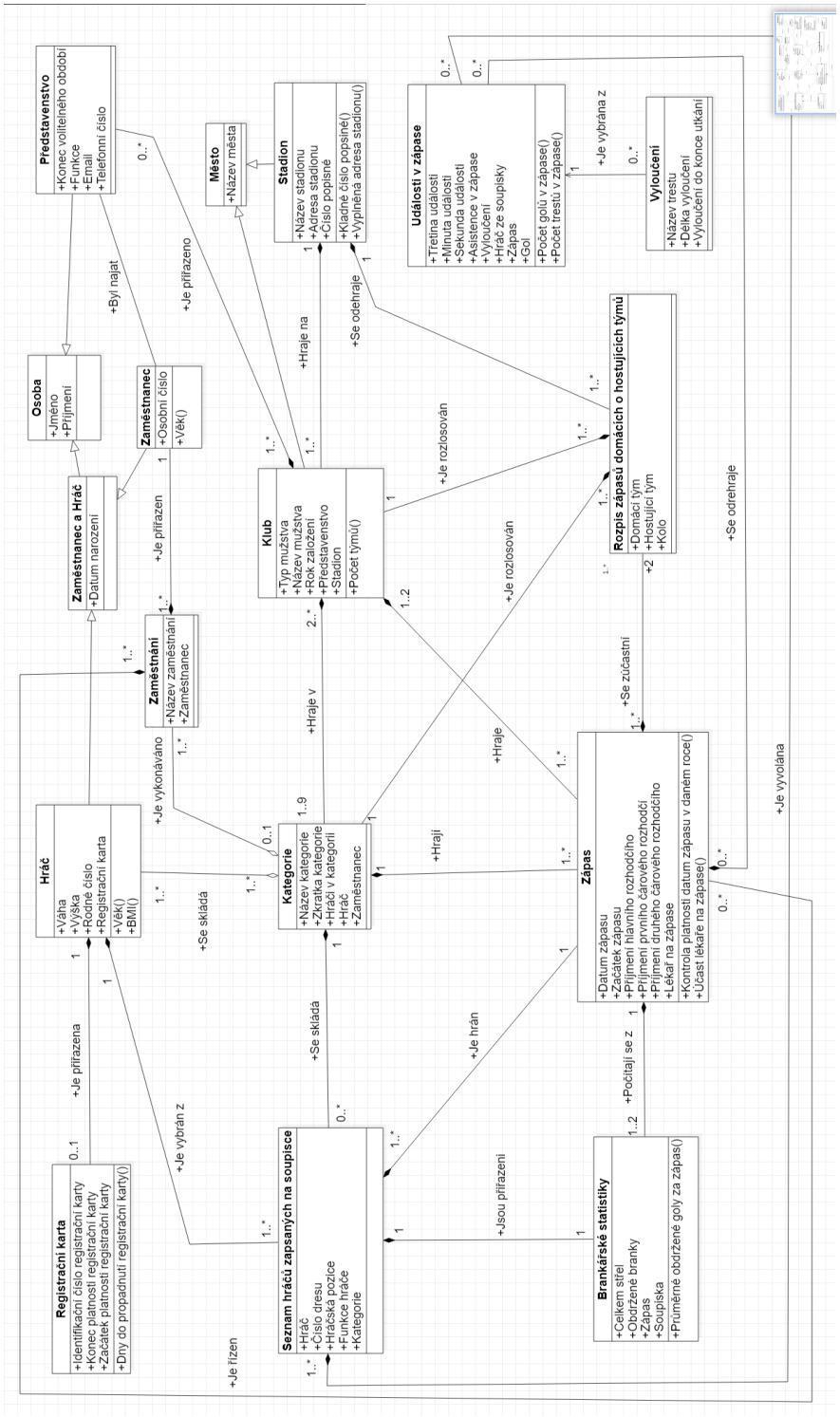
Tresty		
Název trestu	**Délka trestu**	**Trest ve hře**
Rvačka	25	Ano
Naražení na vražení	25	Ano
Naražení na vražení	2	Ne
Faul loktem	2	Ne
Hrubost	2	Ne
Naražení na vražení	25	Ano
Vysoká hůl	10	Ne
Vysoká hůl	2	Ne
Držení	2	Ne
Sekání	2	Ne
Nesportovní chování	2	Ne
Nesportovní chování	10	Ne
Zdržování hry	2	Ne
Hákování	2	Ne
Napadení	2	Ne
Nedovolené bránění	2	Ne
Podražení	2	Ne

Tabulka 15- Tresty, Zdroj: Vlastní zpracování

4.2 Modelování dat

V prvé řadě je zkonstruován obecný návrh databáze pomocí digramu tříd v programu StarUML. V modelování dat je zhotoven Diagram tříd, který byl představen v teoretické

části (kapitola 3.1.5). Model diagramu tříd pro hokejový tým obsahuje 17 tříd. Ten je vytvořen z důvodu návrhu ze kterého vychází relační a objektová databáze.



Obrázek 44- CLASS diagram, Zdroj: Vlastní zpracování

4.3 Relační datový model

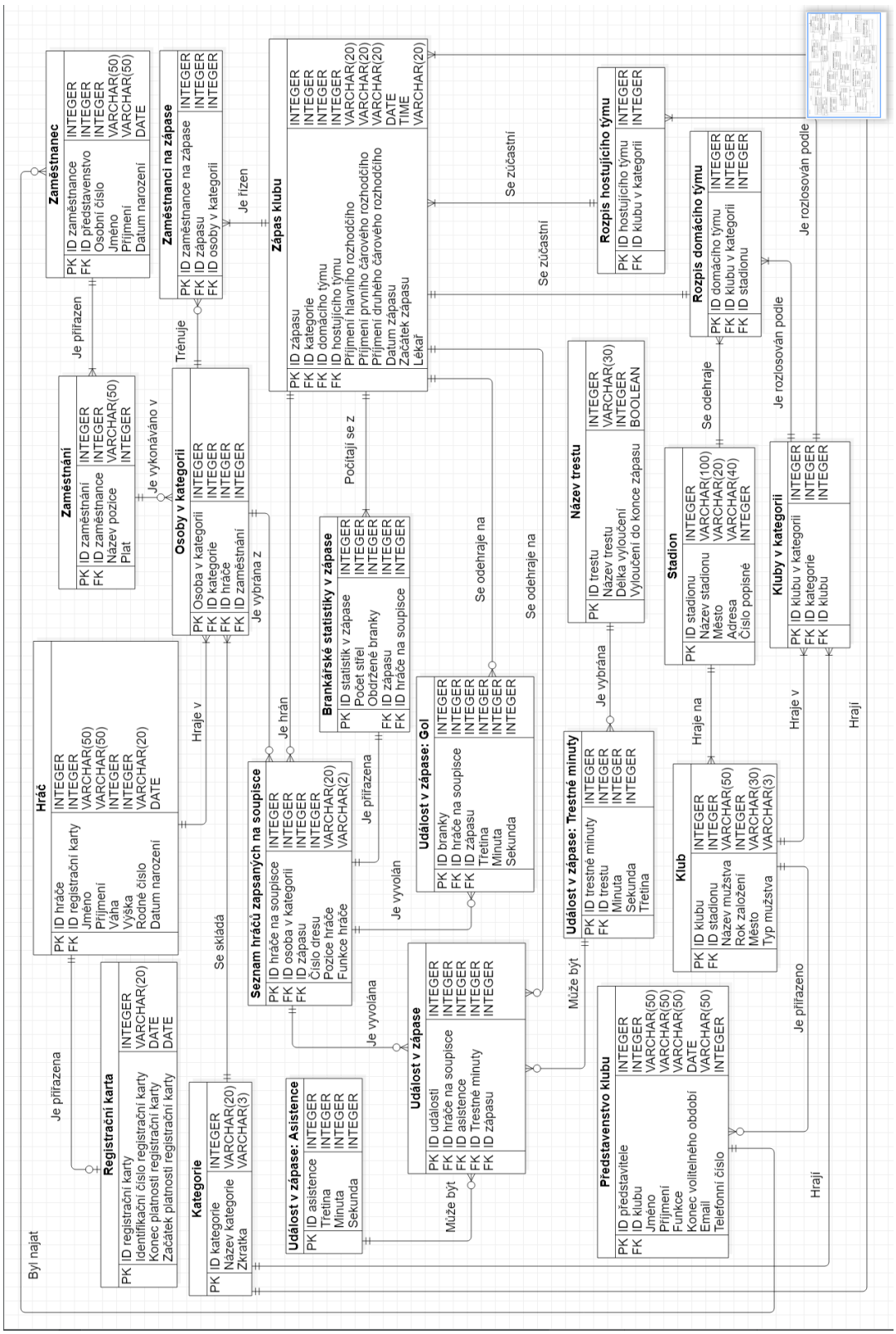
Relační databáze je vytvořena pomocí programu PostgreSQL, který vychází z databázového jazyka SQL. V prvé řadě je zhotoven Implementační E-R diagram, který znázorňuje implementaci relačního modelu s vazebními tabulkami.

4.3.1 E-R diagram

E-R diagram je vytvořen na základě získaných poznatků teoretické části diplomové práce (kapitola 3.1.4). Diagram v této kapitole se lehce liší zobrazením, jelikož nejsou znázorněny vztahy M:N. Z důvodu neschopnosti zobrazení vztahu více k více v relační databázi, je E-R diagram popsán s vazebními tabulkami. Vazební tabulka slouží k složení dvou tříd, ve kterých je vztah M:N.

V modelu jsou nejčastěji zadávány datové typy INTEGER, DATE a VARCHAR (Obrázek 45- Entity Relationship diagram, Zdroj: Vlastní zpracování). VARCHAR ve většině případů je omezený na počet znaků dle zadání modelu. Číselný datový typ INTEGER je použit nejčastěji z důvodu popsání lidských vlastností (výšky, hmotnosti), identifikace času při události v zápase (branka, asistence) a jako jedinečný identifikátor pro relaci. Datový typ DATE se používá v databázi při práci s daty (datum narození, konec platnosti registrační karty).

Třída Osoby v kategorie je vytvořena z důvodu zařazení hráčů a zaměstnanců do jednotlivých kategorií. Z třídy lze následně zapsat hráče na soupisku k jednotlivému zápasu, což představuje třída Seznam hráčské soupisky na zápas.



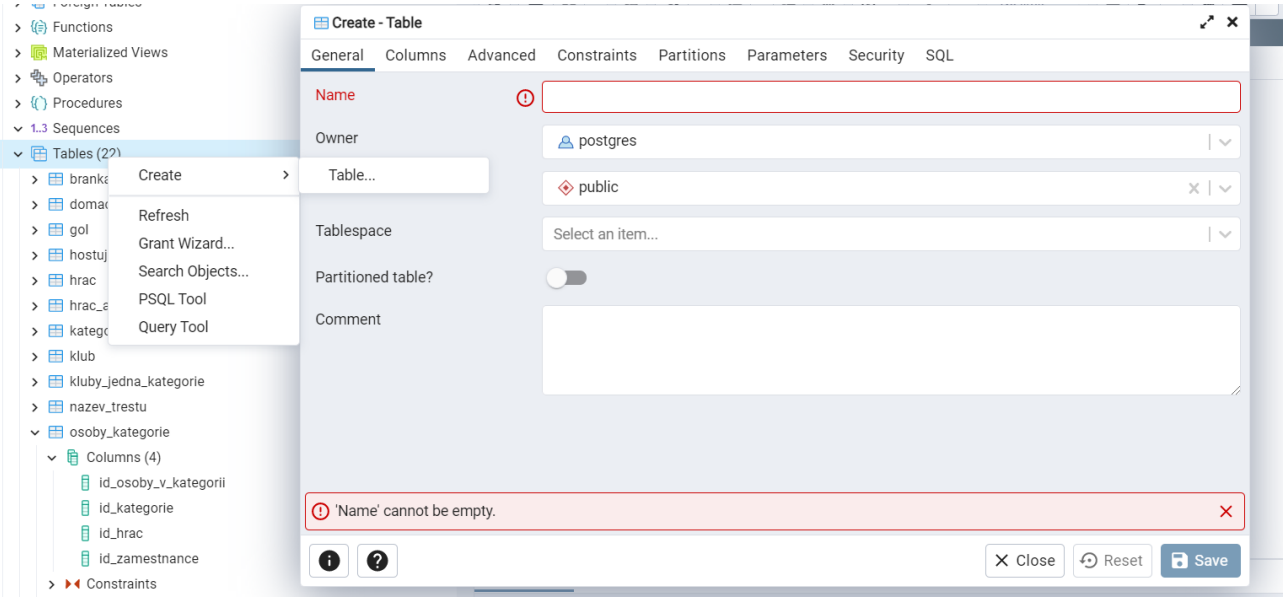
Obrázek 45- Entity Relationship diagram, Zdroj: Vlastní zpracování

4.3.2 Relační databáze

V kapitole Relační databáze je znázorněna ukázka relačního databázového modelu. Relační databázový model vychází z kapitoly v teoretické části Relační databázový model (kapitola 13.2.2). Relační databáze je vytvořena v programu PostgreSQL. PostgreSQL obsahuje mechanické zadávání tabulek a pojmenování atributů u tabulek, které byly použity. Pro příkazy (INSERT, UPDATE…) program PostgreSQL používá textové prostředí (Query Editor). Databáze vychází z E-R diagramu (**Chyba! Nenalezen zdroj odkazů.**).

- Založení tabulky (kapitola 3.2.2.4.5.)

Tabulku lze v programu založit dvěma způsoby. Jeden způsob je pomocí Query Editor, kde založení tabulky lze pomocí syntaxe CREATE TABLE. Autor práce využil druhý způsob, kdy založení tabulky se provede kliknutím pravého tlačítka na link Tables. Poté se uživateli zobrazí tabulka, kde je nutné zadat jméno tabulky (**Chyba! Nenalezen zdroj odkazů.**).

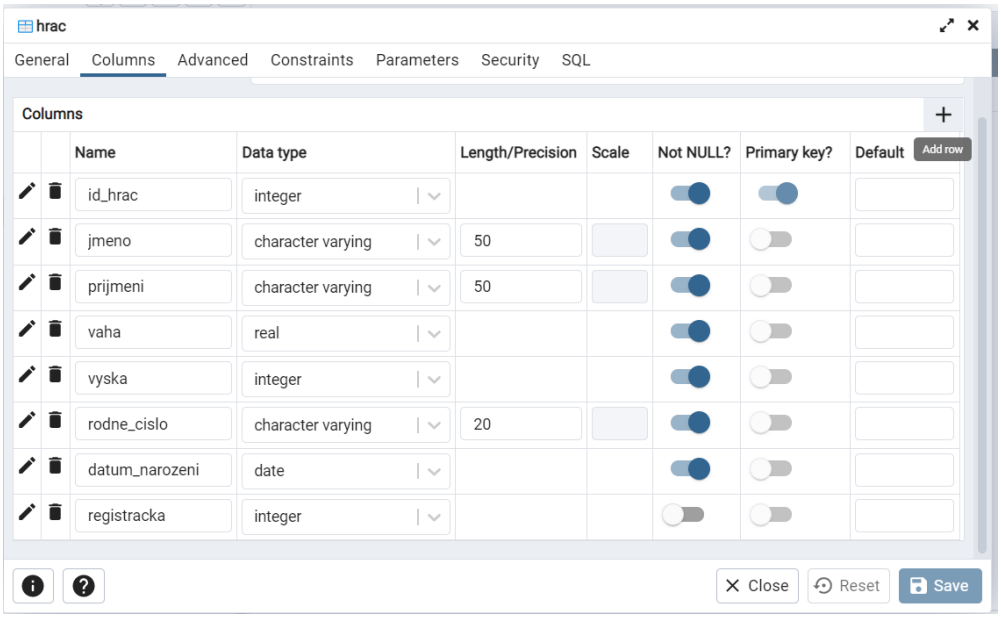


Obrázek 46- Vytvoření tabulky v PostgreSQL, Zdroj: Vlastní zpracování

- Vytvoření a zadání datových atributů pro tabulky (kapitola 3.2.2.1.2)

V záložce Sloupce (Columns) lze pro třídu vytvořit atributy. U atributu se určí jméno atributu, datový typ, maximální délka, zda se jedná o primární klíč, anebo jestli atribut

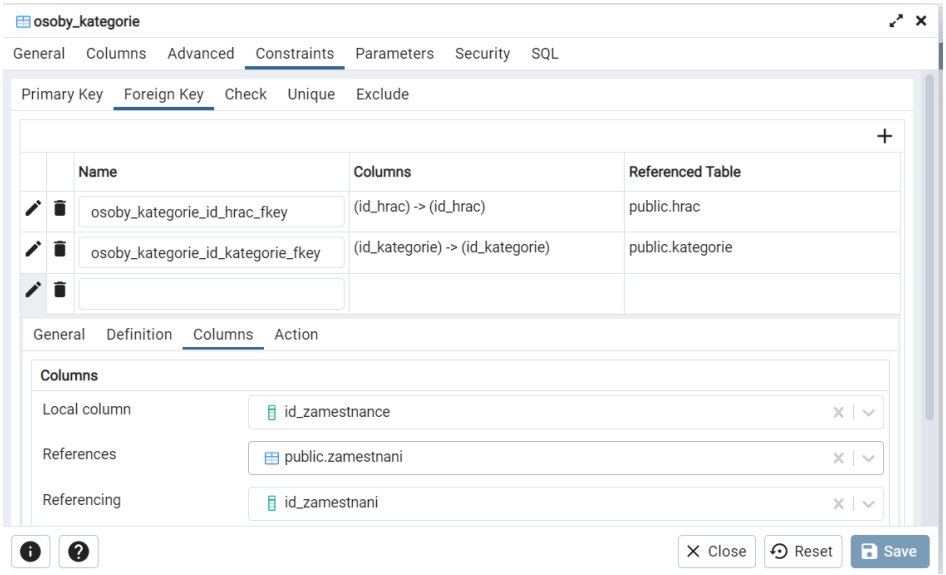
nesmí být nulový. U primárního klíče byl nastaven ještě autoincrement pro generování atributu při přidání nového záznamu.



Obrázek 47-Atributy ve třídě Hráč, Zdroj: Vlastní zpracování

- Propojení cizích klíčů (kapitola 3.2.2.1.4)

Pro připojení cizího klíče mezi tabulkami je nutné v tabulce, kam bude primární klíč odkazován jako cizí, přepnout na záložku Constraints. Poté se přepne na záložku cizí klíč. Následně je nutné vyplnit tři pole. První znamená sloupec, který má být používán jako cizí klíč. Druhý znázorňuje odkaz na ostatní tabulky v databázi a třetí pole je primární klíč, s kterým má být cizí klíč propojen.



Obrázek 48- Cizí klíče ve třídě osoby_kategorie, Zdroj: Vlastní zpracování

- Vkládání dat do tabulek (kapitola 3.2.2.4.5.)

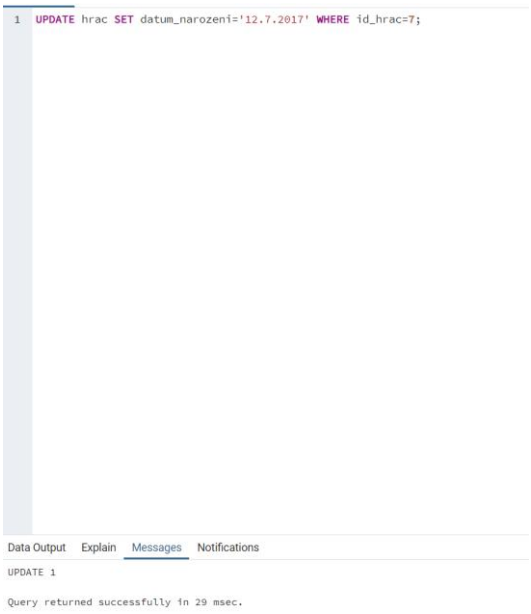
Vložení do tabulek se v databázi používá přes příkaz INSERT INTO. V příkazu INSERT se vypisují id cizích klíčů z důvodu budoucího propojení.

```
INSERT INTO klub
(id_stadionu, rok_zalozeni, mesto, typ_muzstva)
VALUES
(2, 2012,'Písek','HK'),
(3, 1995,'Mirotice', 'HC'),
(4, 1900,'Bludov', 'HK'),
(5, 1985,'Dobřichovice', 'HC'),
(7, 1974,'Kostelec nad Vltavou', 'HK'),
(8, 1937,'Zaječov', 'HC'),
(1, 1895,'Příbram', 'HK'),
(3, 2005,'Blatná','HC'),
(6, 1924,'Český Rudolec','HC'),
(1, 2016,'Hlohovice','HK')
```

Obrázek 49- Vložení dat do třídy Klub, Zdroj: Vlastní zpracování

- Update (viz teoretická část kapitola 3.2.2.4.2)

V projektu se v případě zadání chybných dat nebo doplnění dat do tabulky používal příkaz UPDATE (Obrázek 50- UPDATE u datumu narození). Příkaz UPDATE se používá v Query editor.



Obrázek 50- UPDATE u datumu narození, Zdroj: Vlastní zpracování

- Příkazy (viz teoretická část kapitola 3.2.2.4.1)

Příkazy jsou provedeny dle popisu v teoretické části. V průběhu návrhu databáze bylo provedeno více příkazů SELECT z důvodu kontroly a správného propojení atributů. V následující kapitole jsou zapsány příkazy, které zahrnují nejčastější možnosti pro příkaz SELECT.

- Vypsání všech klubů a stadionů, které se nacházejí v databázi

```
SELECT concat(klub.typ_muzstva,' ',klub.mesto) AS "Jméno klubu", stadion.nazev_stadionu AS "Domácí stadion"
FROM klub
INNER JOIN stadion ON klub.id_stadionu=stadion.id_stadionu;
```

Data Output | Explain | Messages | Notifications

	Jméno klubu text	Domácí stadion character varying (100)
1	HK Písek	U krbu
2	HC Mirotice	Zimní stadion Blatná
3	HK Bludov	Bludovský stadion
4	HC Dobřichovice	Rajec aréna
5	HK Kostelec nad Vltavou	Kostelecký ležák
6	HC Zaječov	Tvrz
7	HK Příbram	Ovál
8	HC Blatná	Zimní stadion Blatná
9	HC Český Rudolec	Zimní stadion v Telči
10	HK Hlohovice	Ovál

Obrázek 51-Vypsání klubů se stadiony, Zdroj: Vlastní zpracování

- Vypsání soupisky hráčů A-Týmu proti HC Zaječov

```
SELECT hrac.jmeno, hrac.prijmeni, zapsani_hrace_soupiska.cislo_dresu, zapsani_hrace_soupiska.funkce_hrace
FROM zapsani_hrace_soupiska
JOIN osoby_kategorie ON osoby_kategorie.id_osoby_v_kategorii= zapsani_hrace_soupiska.id_osoba_kat
JOIN hrac ON hrac.id_hrac= osoby_kategorie.id_hrac
JOIN zapas_klubu ON  zapsani_hrace_soupiska.id_zapas= zapas_klubu.id_zapasu
JOIN kategorie ON kategorie.id_kategorie= osoby_kategorie.id_kategorie
JOIN domacitym ON domacitym.id_domaciho_tymu= zapas_klubu.domaci_tym
JOIN kluby_jedna_kategorie ON kluby_jedna_kategorie.id_klubu_v_kategorii= domacitym.id_klubuvkategorii
JOIN klub ON klub.id_klubu = kluby_jedna_kategorie.id_klubu
WHERE kategorie.nazev_kategorie='A-tým' AND klub.mesto='Zaječov' AND klub.typ_muzstva='HC'
```

Data Output | Explain | Messages | Notifications

	jmeno character varying (50)	prijmeni character varying (50)	cislo_dresu integer	funkce_hrace character varying (2)
1	Martin	Ruml	15	[null]
2	Šimon	Král	20	[null]
3	Mikuláš	Vráblík	52	[null]
4	Karel	Němec	12	[null]
5	Stanislav	Pokorný	2	[null]
6	Tomáš	Němý	3	[null]
7	Robert	Krysa	13	[null]
8	Aleš	Klobása	1	[null]
9	Miroslav	Červený	39	[null]
10	Ivan	Knotek	10	C
11	Michal	Tomáš	4	A1
12	Boris	Stehno	41	A2

Obrázek 52- Soupiska A-Týmu proti HC Zaječov (Relační databáze, Zdroj: Vlastní zpracování)

- Zaměstnanci u A-týmu – Z důvodu většího počtu zaměstnanců byly vypsány pouze zaměstnanci, kteří se nachází u A-týmu

```
SELECT l.jmeno, l.prijmeni, v.nazev_pozice
FROM osoby_kategorie j, kategorie f, zamestnani v, zamestnanec l
WHERE j.id_kategorie = f.id_kategorie AND j.id_zamestnance = v.id_zamestnani AND v.id_zamestnance = l.id_zamestnance AND f.nazev_kategorie='A-tým'
```

Data Output | Explain | Messages | Notifications

	jmeno character varying (50)	prijmeni character varying (50)	nazev_pozice character varying (50)
1	Čeněk	Masař	Hlavní trenér A-tým
2	Tomáš	Pecha	Asistent trenéra A-tým
3	Oto	Vorlíček	Vedoucí mužstva A-tým

Obrázek 53- Vypsání zaměstnanců u A-týmu pomocí tečkové notace, Zdroj: Vlastní zpracování

- Vypsání klubů v kategorii Nábor se stadionem

```
SELECT concat (klub.typ_muzstva,' ',klub.nazev_muzstva, ' ', klub.mesto), stadion.nazev_stadionu
FROM kluby_jedna_kategorie
LEFT JOIN kategorie ON kategorie.id_kategorie = kluby_jedna_kategorie.id_kategorie
INNER JOIN klub ON kluby_jedna_kategorie.id_klubu = klub.id_klubu
INNER JOIN stadion ON stadion.id_stadionu = klub.id_stadionu
WHERE kategorie.nazev_kategorie='Nábor'
```

Data Output | Explain | Messages | Notifications

	concat text	nazev_stadionu character varying (100)
1	HK Lvi Příbram	Ovál
2	HK Písek	U krbu
3	HC Žraloci Blatná	Zimní stadion Blatná
4	HK Bludov	Bludovský stadion
5	HK Kostelec nad Vltavou	Kostelecký ležák

Obrázek 54- Kluby v kategorii Nábor, Zdroj: Vlastní zpracování

- Vypsání hráčů klubu včetně registrační karty (hráči kategorie Nábor nemusí mít)

```
SELECT hrac.jmeno, hrac.prijmeni,hrac.rodne_cislo, registracni_karta.ident_cislo
FROM hrac
RIGHT JOIN registracni_karta ON registracni_karta.id_registracky = hrac.registracka
LIMIT 10
```

Data Output | Explain | Messages | Notifications

	jmeno character varying (50)	prijmeni character varying (50)	rodne_cislo character varying (20)	ident_cislo character varying (20)
1	Jonáš	Brzák	150805/1654	S450100
2	Vlastimil	Mlčák	140907/1174	S450101
3	Kryštof	Vyroubal	140801/1074	S450102
4	Daniel	Kopeček	120203/1421	S450103
5	Zdeněk	Tesař	120124/1111	S450104
6	Erik	Janečka	111108/1453	S450105
7	Ladislav	Šedivý	110405/1865	S450106
8	Vít	Kulík	110401/1762	S450107
9	Rostislav	Šmerda	110703/1234	S450108
10	Ivan	Čapek	110805/1398	S450109

Obrázek 55- Vypsání prvních dvaceti hráčů klubu s platnou registrační kartou

- Představenstvo klubu HC Lvi Příbram

```
SELECT predstavenstvo.jmeno, predstavenstvo.prijmeni, predstavenstvo.funkce, predstavenstvo.email,predstavenstvo.telefonni_cislo
FROM predstavenstvo
JOIN klub ON predstavenstvo.id_klubu= klub.id_klubu
WHERE klub.nazev_muzstva='Lvi' AND klub.mesto='Příbram' AND klub.typ_muzstva='HC'
```

Data Output | Explain | Messages | Notifications

	jmeno character varying (50)	prijmeni character varying (50)	funkce character varying (50)	email character varying (50)	telefonni_cislo integer
1	Jonáš	Černík	Prezident klubu	jonas.cermak@seznam.cz	414514321
2	Soběslav	Kožíšek	Sportovní manažer	sobeslav.kozisek@lvipribram.cz	488888888
3	Marek	Zavadil	Vedoucí mládeže	zavadil@lvipribram.cz	111888687
4	Jana	Svátková	Marketingová ředitelka	j.svatkova@lvipribram.cz	547880000
5	Kateřina	Kolová	Finanční ředitelka	k.kolova@lvipribram.cz	555555555

Obrázek 56- Představenstvo klubu, Zdroj: Vlastní zpracování

- Příklad pro vypočtení metody bylo vybráno vypočtení konce platnosti registrační karty u hráčů v klubu.

```
SELECT hrac.jmeno, hrac.prijmeni, konec_platnosti- current_date AS "Dny do konce registracni karty"
FROM  hrac
RIGHT JOIN registracni_karta ON hrac.registracka = registracni_karta.id_registracky ORDER BY konec_platnosti
```

Data Output | Explain | Messages | Notifications

	jmeno character varying (50)	prijmeni character varying (50)	Dny do konce registracni karty integer
1	Jana	Sedláčková	298
2	Dominik	Kreslený	303
3	Dalimij	Blažek	320
4	Erik	Janečka	345
5	Šimon	Král	346
6	Ivan	Čapek	352
7	Mojmír	Borovička	361
8	Vilém	Kotala	387
9	Daniel	Kopeček	414
10	Emil	Holomek	419

Obrázek 57- Počet dnů do konce platnosti registračky (relační databáze), Zdroj: Vlastní zpracování

4.4 Objektový datový model

Objektová databáze je vytvořena pomocí programu Daskalos. Objektová databáze vychází z Objektově databázový systému, který byl popsán v teoretické části diplomové práce. Pro vkládání dat používá program Daskalos záložku script. Výhoda programu spočívá v možnosti zobrazení diagramu s propojením jednotlivých tříd v projektu.

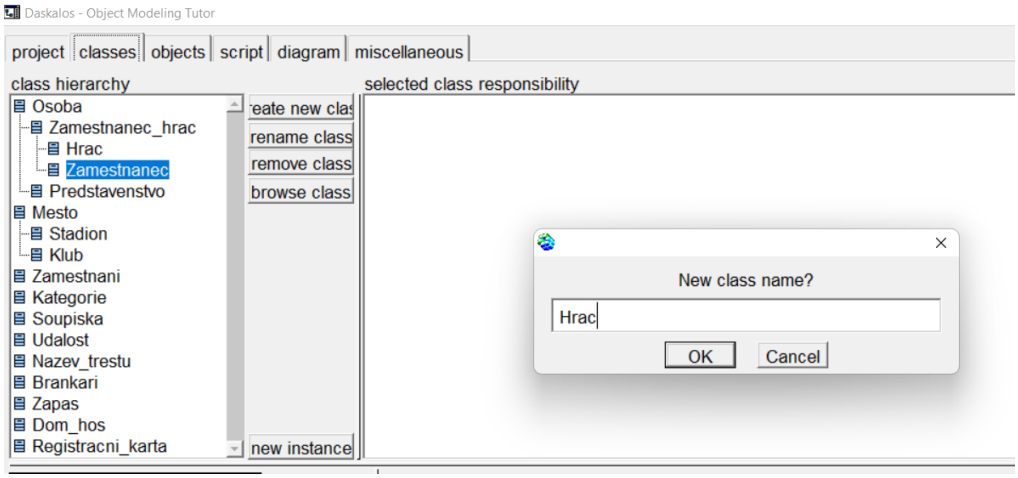
4.4.1 Diagram tříd

V Diagram tříd vychází z diagramu tříd. (4.2). Jediný rozdíl je možnost zobrazit jednotlivá propojení.

4.4.2 Objektová databáze

- Založení objektu

Objekt v objektové databázi Daskalos se založí pomocí záložky třídy (classes) na horní liště. V záložce se nová třída vytvoří pomocí tlačítka create new class, kde se následně zadá jméno třídy, která je nutná vytvořit.

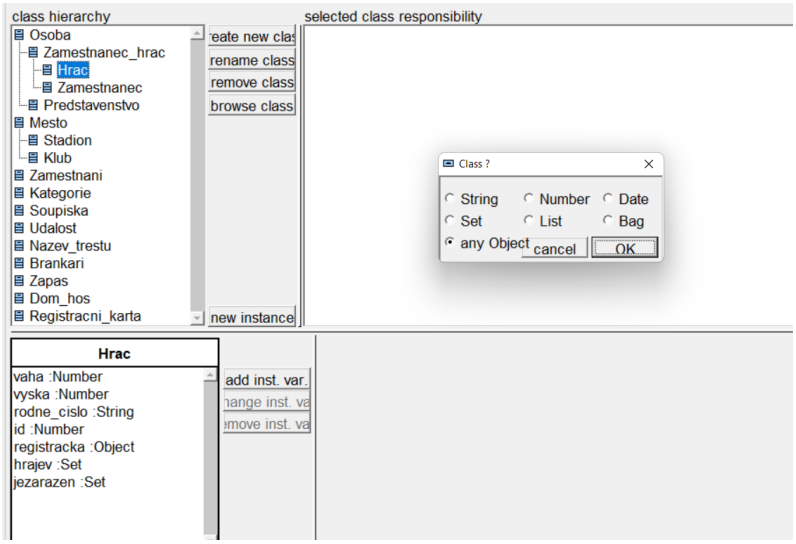


Obrázek 58- Vytvoření třídy v objektové databázi Daskalos, Zdroj: Vlastní zpracování

- Vytvoření a zadání datových atributů pro Objekt Hráč

Instance se v objektové databázi přidávají na stejné liště jako vytvoření nové třídy. Při vytvoření nové instance je nutné nejdříve napsat název instance a poté se vybere datový typ instance. Pro vytvoření klasické instance lze použít číselný typ (Number), textový řetězec (String) a datum (DATE).

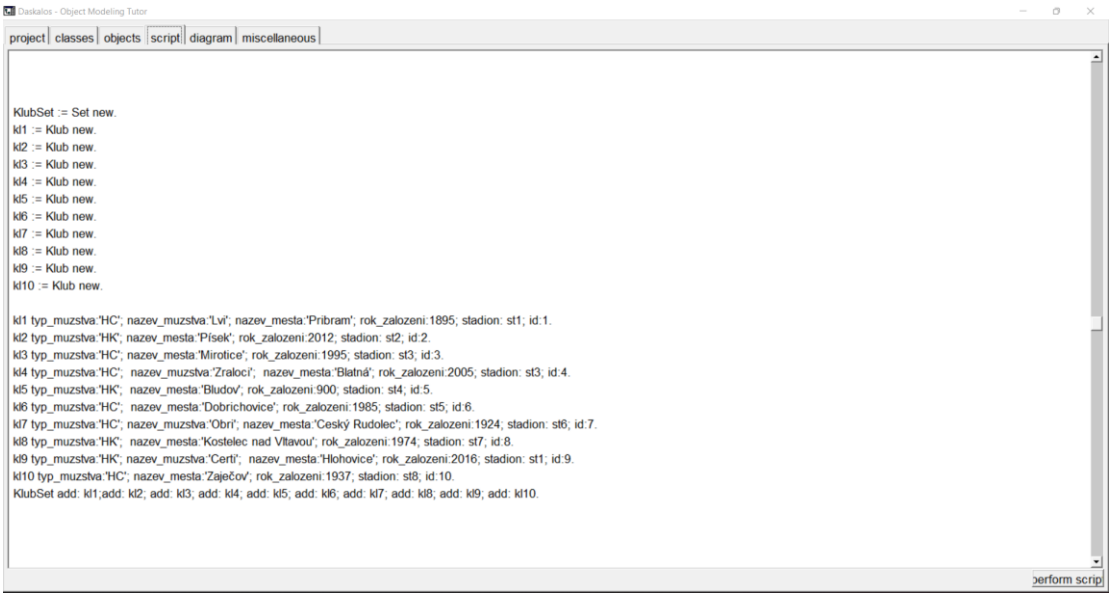
Datové typy Set, List, Bag a any Object a budou využity při propojení mezi objekty dle vztahu (kapitola 3.2.3).



Obrázek 59- Výběr datového typu v objektové databázi Daskalos, Zdroj: Vlastní zpracování

- Vkládání dat do objektu Hráč

Pro vložení dat je používána záložka script. Přiřazení objektu k třídě Klub se provádí pomocí naz_objektu(kl1) := Klub new. Data do objektu se vkládají pomocí syntaxe jmeno_instance:informace. Datový typ String se zadává pomocí 'text', číslo se zadává bez speciálního znaku a datum podle syntaxe 'MM-DD-RRRR' (měsíc- den-rok). Data se uloží po kliknutí tlačítka perform script, který se nachází na pravé straně dolní části programu.



Obrázek 60- Vložení dat do třídy Klub, Zdroj: Vlastní zpracování

- Propojení tříd objektů

Při propojení tříd mezi sebou se v prvé řadě musí správně určit, o jaký vztah mezi třídami se jedná. Při vztahu M:N se používají datové typy Set, Bag a List, při vztahu 1:N se používá datový typ Object (viz kapitola Objektově databázový systém). Název instance je znázorněn na přímce mezi propojenými třídami.

Vztah 1:N se přiřadí do scriptu při zadávání dat do instance. Objekt se vztahem 1 se propojí přes Object při zadávání dat do objektu. Při vkládání musí platit podmínka o předešlém definování objektu.

```
kl1 typ_muzstva:'HC'; nazev_muzstva:'Lvi'; nazev_mesta:'Pribram'; rok_zalozeni:1895; stadion: st1; id:1.
kl2 typ_muzstva:'HK'; nazev_mesta:'Písek'; rok_zalozeni:2012; stadion: st2; id:2.
kl3 typ_muzstva:'HC'; nazev_mesta:'Mirotice'; rok_zalozeni:1995; stadion: st3; id:3.
kl4 typ_muzstva:'HC';  nazev_muzstva:'Zraloci';  nazev_mesta:'Blatná'; rok_zalozeni:2005; stadion: st3; id:4.
kl5 typ_muzstva:'HK';  nazev_mesta:'Bludov'; rok_zalozeni:900; stadion: st4; id:5.
kl6 typ_muzstva:'HC';  nazev_mesta:'Dobrichovice'; rok_zalozeni:1985; stadion: st5; id:6.
```

Obrázek 61- Propojení stadionu ke klubu v objektové databázi, Zdroj: Vlastní zpracování

Ve vztahu M:N musí také propojovaný objekt již existovat. Příklad pro propojení více objektů je přidání klubů v kategorii do databáze pomocí typu Set. Při zadání musí být použito pomocné slovo add a objekty odděleny středníkem.

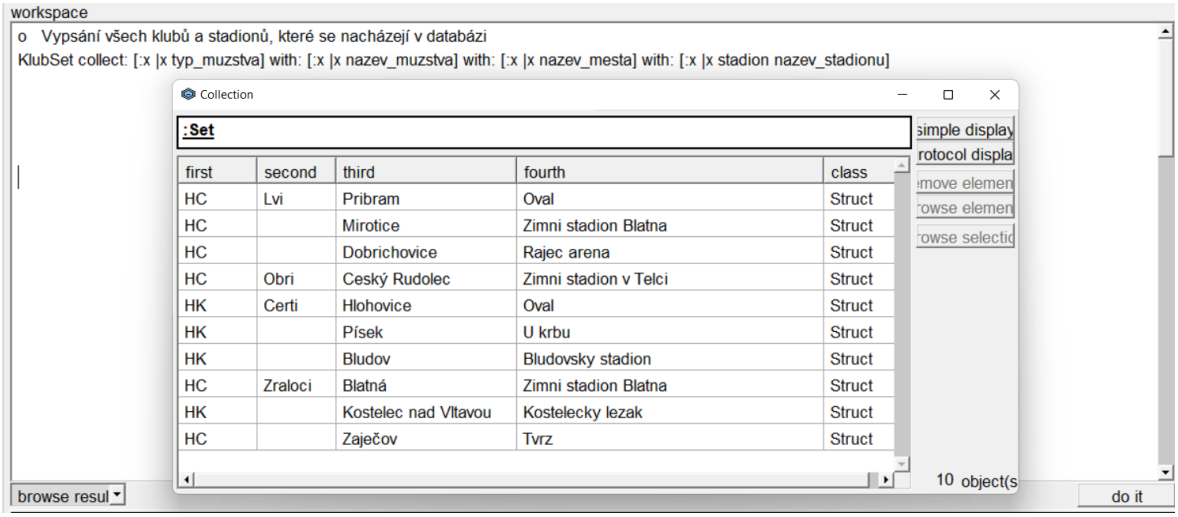
```
"Pridani klubu do kategorie"
kat1 tymy add: kl1; add: kl2; add: kl4;add: kl5; add: kl8.
kat2 tymy add: kl1; add: kl3; add: kl7; add: kl8.
kat3 tymy add: kl1; add: kl3; add: kl6; add: kl7; add: kl8; add: kl9.
kat4 tymy add: kl1; add: kl5; add: kl9; add: kl10.
kat5 tymy add: kl1; add: kl2; add: kl3; add: kl4; add: kl5; add: kl6; add: kl7; add: kl8.
kat6 tymy add: kl1;  add: kl2; add: kl3; add: kl4; add: kl5; add: kl6; add: kl7; add: kl8.
kat7  tymy add: kl1; add: kl3; add: kl5; add: kl7.
kat8 tymy add:kl1; add: kl2;  add: kl6;  add: kl9;  add: kl10.
kat9 tymy add: kl1;  add: kl2;  add: kl5;  add: kl8;  add: kl10.
```

Obrázek 62- Přiřazení klubů do kategorií, Zdroj: Vlastní zpracování

- Aktualizování dat

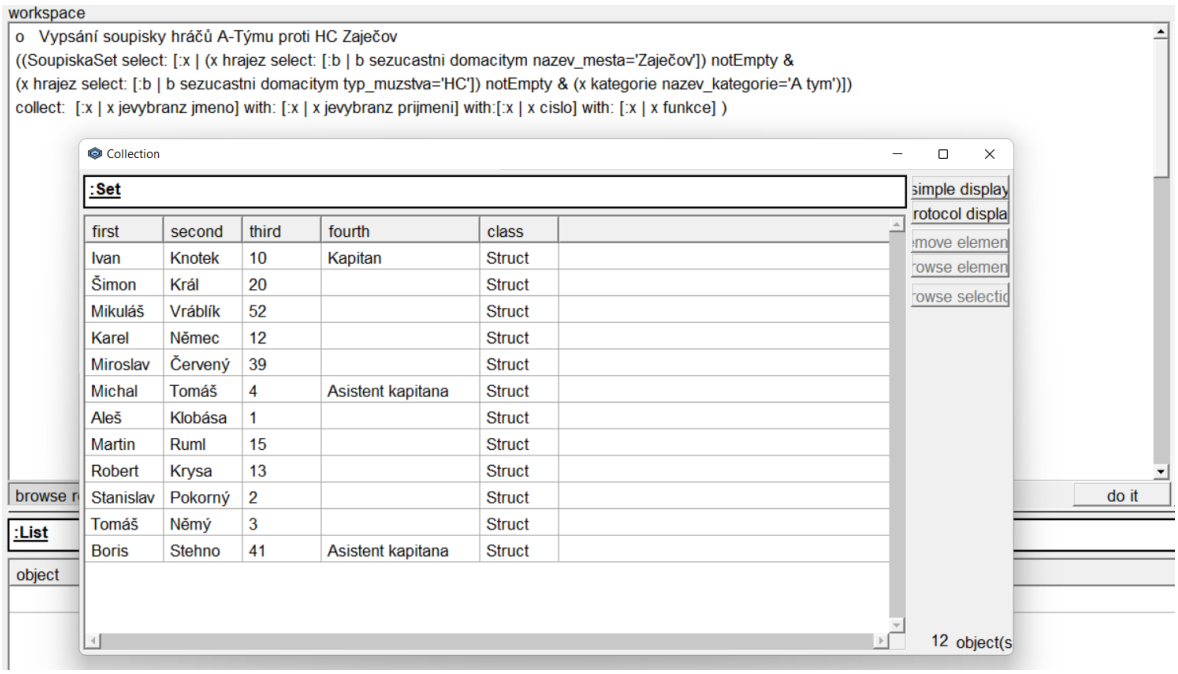
Aktualizace dat objektové databáze je jednoduchá. Aktualizace se nachází v záložce script, kde se najde příslušný řádek aktualizace. Po aktualizaci se výsledek uloží pomocí tlačítka perform script.

- Příkazy (viz kapitola 3.2.3.3.2
    - Vypsání všech klubů a stadionů, které se nacházejí v databázi



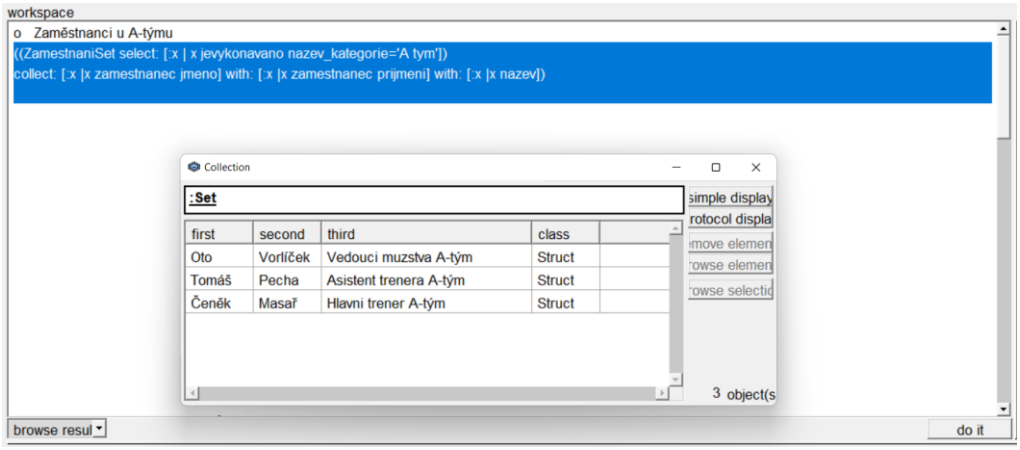
Obrázek 63-Vypsání klubů se stadiony, na kterých hraje v relační databázi, Zdroj: Vlastní zpracování

    - Vypsání soupisky hráčů A-Týmu proti HC Zaječov



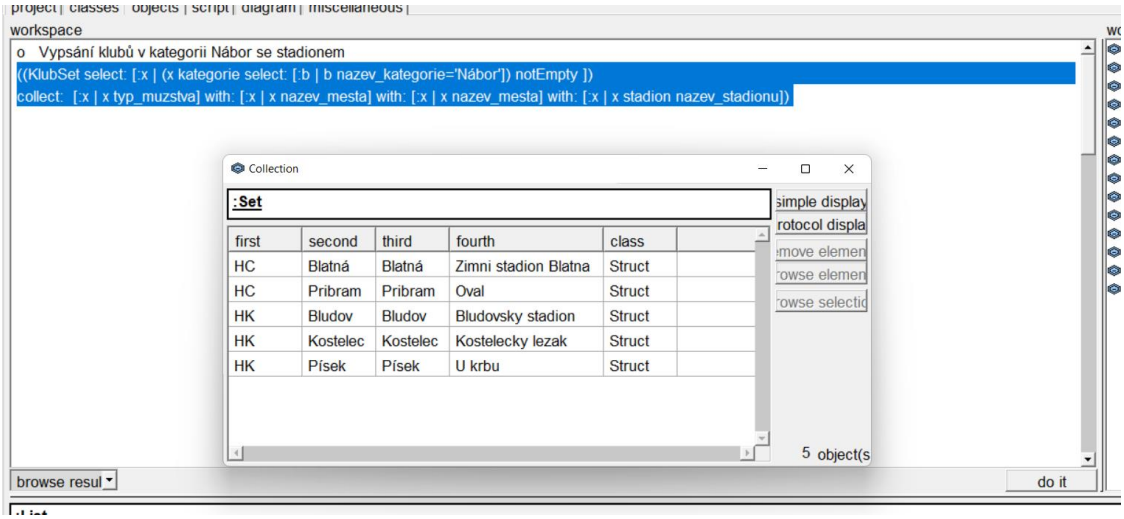
Obrázek 64- Vypsání hráčů A-týmu v zápase proti HC Zaječov, Zdroj: Vlastní zpracování

- Zaměstnanci u A-týmu – Z důvodu většího počtu zaměstnanců byly vypsány pouze zaměstnanci, kteří se nachází u A-týmu



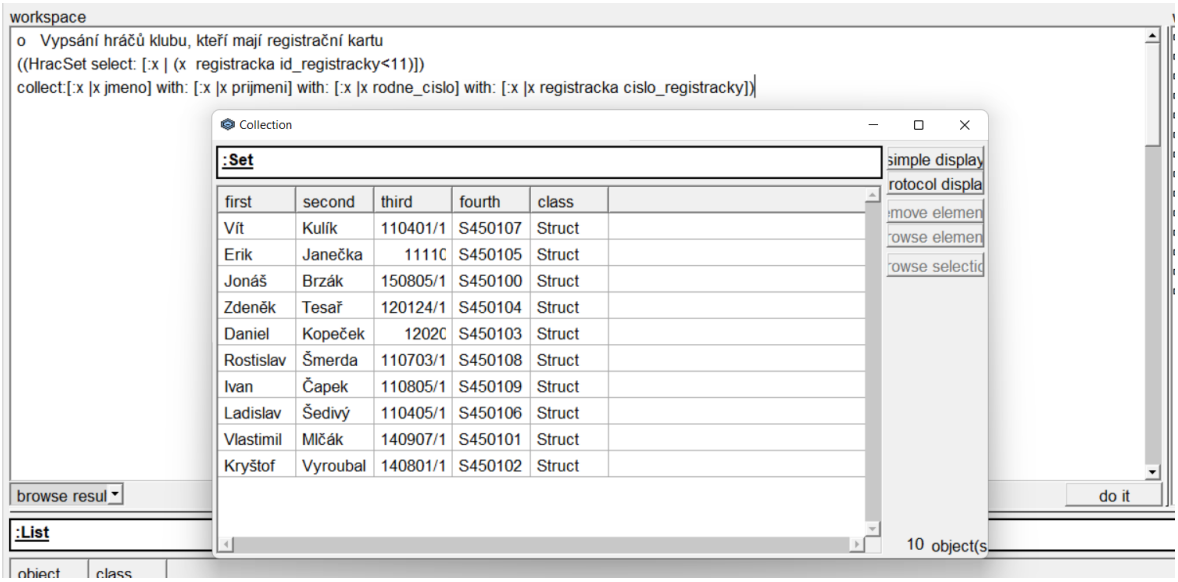
Obrázek 65- Vypsání zaměstnanců A-týmu v objektové databázi, Zdroj: Vlastní zpracování

- Vypsání klubů v kategorii Nábor se stadionem



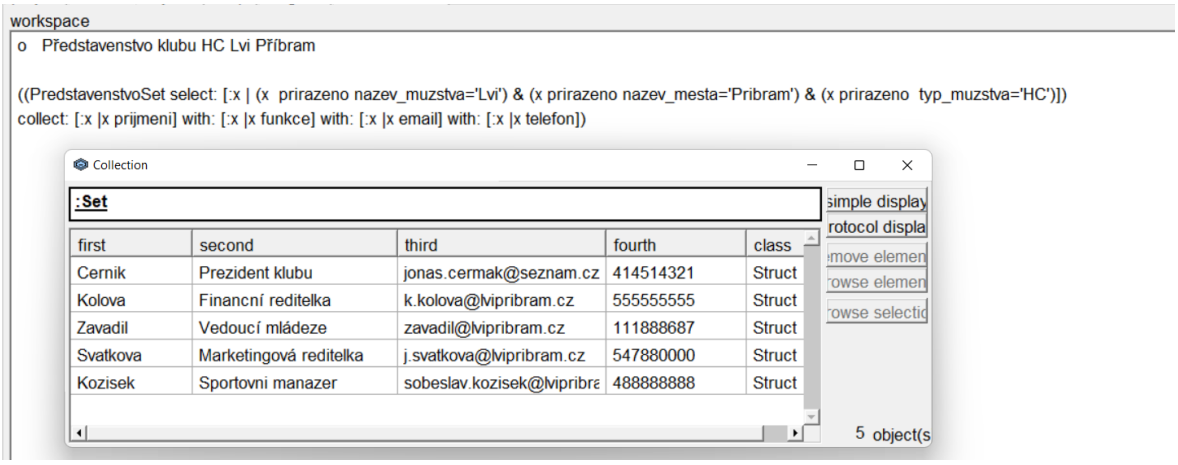
Obrázek 66- Kluby a stadiony, kteří hrají v kategorii Nábor, Zdroj: Vlastní zpracování

- Vypsání hráčů klubu, kteří mají registrační kartu (hráči kategorie Nábor nemusí mít)



Obrázek 67- Vypsání prvních deset hráčů s registrační kartou v objektové databázi, Zdroj: Vlastní zpracování

- Představenstvo klubu HC Lvi Příbram



Obrázek 68- Představenstvo klubu pomocí objektové databáze, Zdroj: Vlastní zpracování

- Příklad pro výpočet konce platnosti registrační karty u hráčů v klubu.

```
method code:
dobadokonceplatnosti
    konec isNil ifTrue: [^0].
    ^(Date today subtractDate: konec) * -1 truncated
```

Obrázek 69- Metoda pro vypočtení času do konce platnosti registrační karty v objektové databázi, Zdroj: Vlastní zpracování

4.5 Dotazníkové šetření

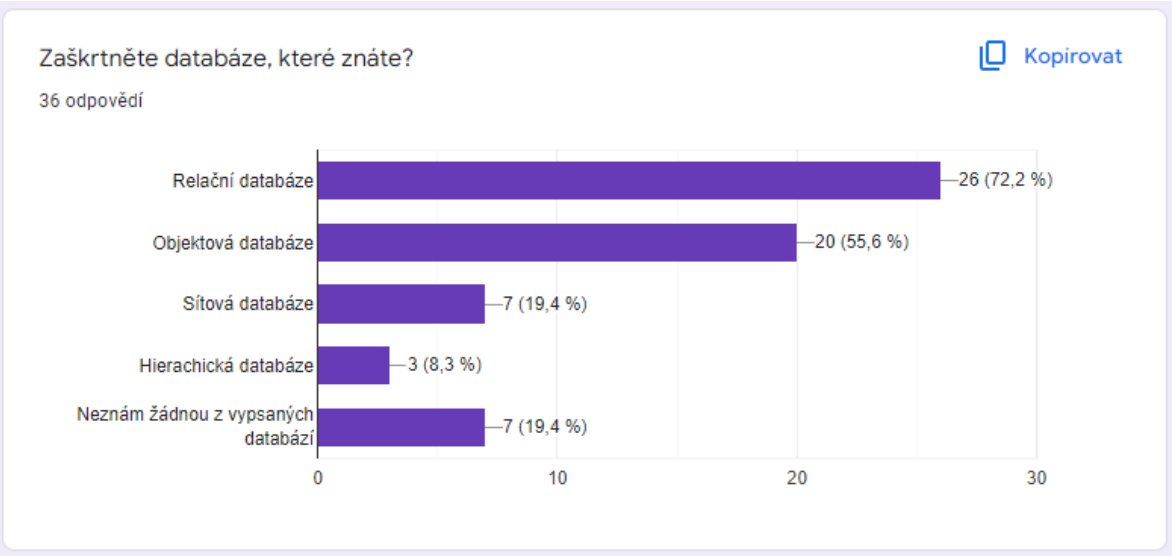
Po implementačním řešení v praktické části jsou obě databáze porovnány pomocí dotazníku. Otázky v dotazníku byly vytvořeny pomocí parametrů, které vycházejí z normy ČSN ISO/IEC 25000. Otázka je vždy položena na podobném principu pro objektovou i relační databázi. V případě práce s daty byly úkoly náročně podobné. Zadané úkoly se lišily z důvodu rozlišení a nemožnosti udělat „jedním“ krokem dvě otázky v dotazníku najednou. Korespondentům byla poskytnuta část databáze s potřebnými daty pro splnění jednotlivých úkolů. Dotazník byl vytvořen v online nástroji, který nabízí účet na Googlu. Dotazník byl vytvořen za pomocí dat (kapitole 4.1).

4.5.1 Výsledky Dotazníkového šetření

Výsledky dotazníkového řešení jsou znázorněny pomocí grafů. U každého grafu je vyhodnocení. Dotazníkového šetření se zúčastnilo celkem 36 korespondentů. Ve většině případů se jedná o osoby, které již někdy pracovaly s některým typem testované databáze. Při úkolech z relační databáze nebyl omezen způsob zpracování. V objektové databázi byl požadován program Daskalos, který slouží pro školní účely.

Dotazník obsahoval otázku na znalost jednotlivých databází. Jednalo se o otázku, kdy korespondent mohl zaškrtnou více odpovědí najednou. Největší povědomost je o relační databázi s 72,2 %, která je v praxi také nejvíce používána. Na druhém místě se umístila objektová databáze s více jak 55 %. O starších databázových modelů, síťová

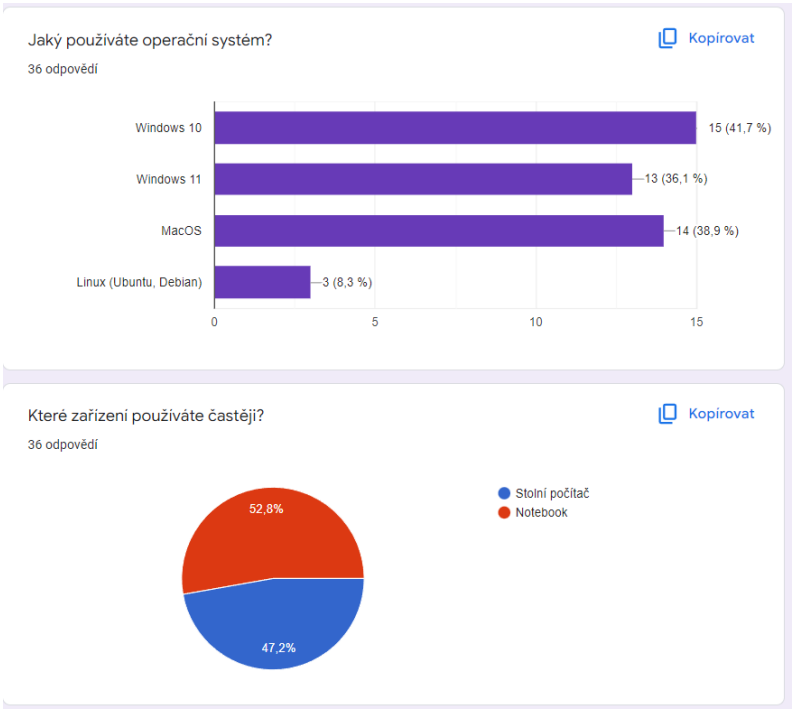
a hierarchická, vědělo 3 a 7 dotázaných. O žádné databázi nemá povědomí 7 korespondentů.



Obrázek 70- Povědomí, jaké druhy databáze existují, Zdroj: Vlastní zpracování

V první sekci dotazníku byly zjištěny operační systémy a typy zařízení, kteří korespondenti používají. Otázky byly zvoleny z důvodu, že kdyby při instalaci konkrétního databázového systému nastal nějaký problém, mohl by se vyhodnotit ke konkrétnímu operačnímu systému.

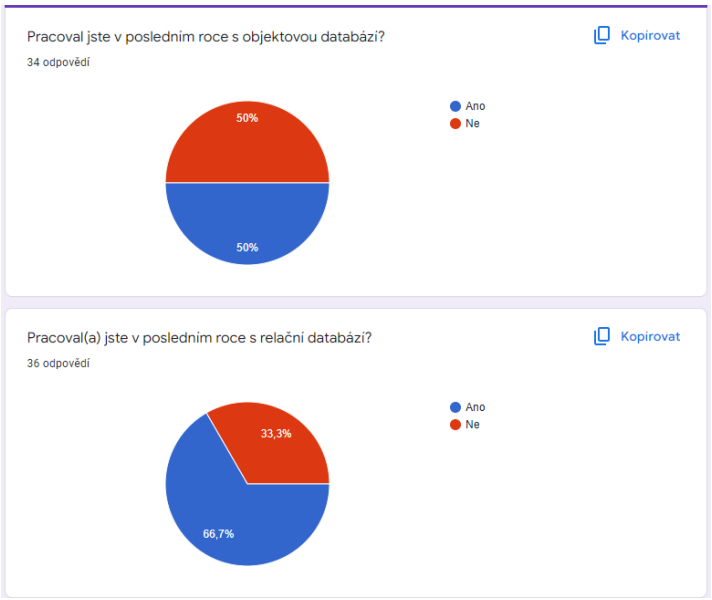
Nejčastěji byly vybrány korespondenty nejoblíbenější operační systémy Windows 10 a MacOS. Následoval Windows 11, který se nyní řadí jako nejnovější operační systém.



Obrázek 71- Zjištění obecných informací o systémech korespondentů v parametru Instalovatelnost, Zdroj: Vlastní zpracování

Pro srovnatelné výsledky dotazníku byla na začátku otázka ohledně databáze, zda korespondent pracoval v posledním roce s relační nebo objektovou databází. Tyto otázky byly zadány z důvodu vyhodnocení výsledků dotazníku, aby výsledky nebyly zkresleny z důvodu neznalosti.

Korespondenti se častěji přihlásili pro práci k relační databázi. Jelikož je více známá, tak s ní se mohou korespondenti častěji dostat do kontaktu například přes backend webů. Další otázky byly voleny dle parametrů kvality dle ISO/IEC 25000.



Obrázek 72- Vybrání korespondentů pro otázky, kteří v posledním roce pracovali s objektovou a relační databází, Zdroj: Vlastní zpracování

4.5.1.1 Parametr Instalovatelnost

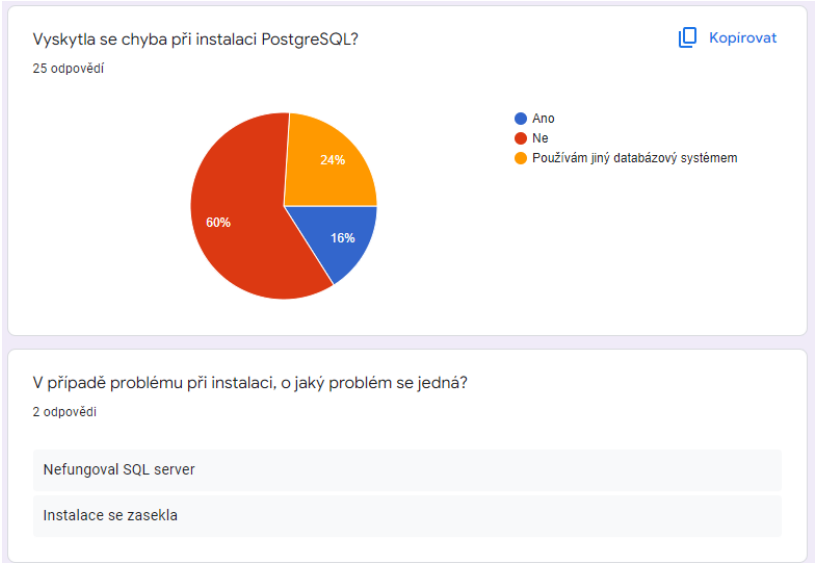
Parametr vychází z možnosti, jak lze systém instalovat do prostředí OS. Při vyšším počtu problému s instalací lze přiřadit instalace ke konkrétnímu operačnímu systému. U relační databáze PostgreSQL je možné stažení zdarma. Pro relační databázi byl použit zdroj https://www.enterprisedb.com/downloads/postgres-postgresql-downloads. Objektová databáze je ke stažení zdarma ze zdroje https://sites.google.com/site/daskalosapplication/instalace/registrace. V objektové databázi je nutná registrace se zadáním jména a emailové adresy.

Otázka v dotazníku má jiný počet odpovědí pro obě databáze z důvodu vyplnění jen korespondenty, kteří mají zkušenosti s daným typem databáze.

- Relační databáze

  V relační databázi byla pro korespondenty možnost zadat odpověď, zda využívají jiný databázový systém. Této možnosti využilo 6 korespondentů. Zbylých 19 mělo určit, zda se při instalaci vyskytla nějaká chyba. Potíže zaznamenali 4 korespondenti, dva měli problém při instalaci. Nalezené problémy se týkaly nefunkčnosti SQL serveru a zaseknutí systému při instalaci. Jelikož problém nastal na dvou odlišných systémech, tak byl tento problém vyhodnocený jako ojedinělý.

Obrázek 73- Parametr Instalovatelnost pro relační databázi



Obrázek 74- Parametr Instalovatelnost pro relační databázi, Zdroj: Vlastní zpracování

- Objektová databáze

V objektové databázi byla otázka celkem zodpovězena sedmnáctkrát. Čtrnáctkrát proběhla instalace v pořádku. V objektové databázi jeden korespondent nenapsal důvod, proč byl problém při instalaci. Ostatní důvody se týkaly registrace a neotevření souboru xml. Ostatní korespondenti buď měli již nainstalovaný program, anebo se bez problémů registrovali.

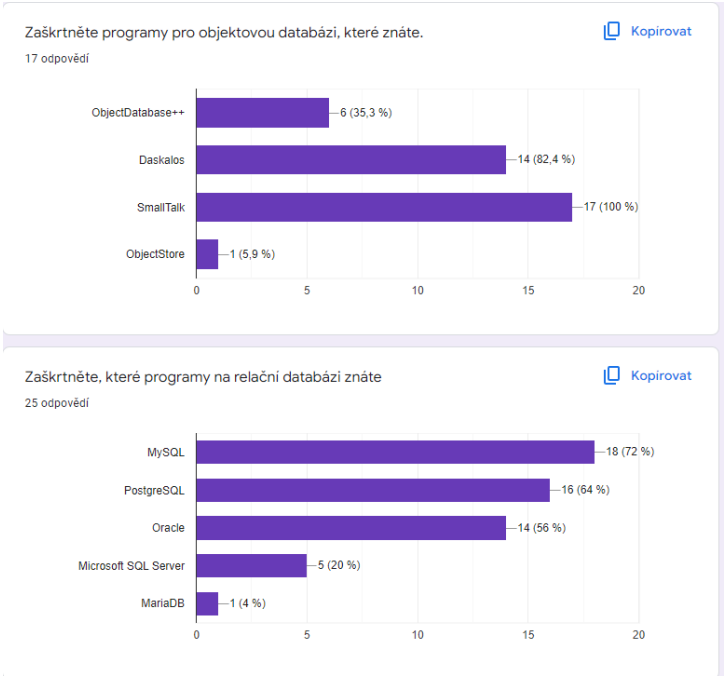
Obrázek 75- Parametr instalovatelnost pro objektovou databázi, Zdroj: Vlastní zpracování

4.5.1.2 Parametr Nahraditelnosti

Parametr nahraditelnosti byl v dotazníkovém šetření zjištěn pomocí znalosti korespondentů, jaké databázové modely znají. Byly vybrány nejznámější modely pro obě databáze.

V Obrázek 76- Parametr Nahraditelnosti) jsou znázorněny výsledky. Na objektovou databázi odpovědělo 17 korespondentů a na relační databázi 25.

Rozdíl v relační a objektové databázi je způsoben, že databázových systémů pro objektovou databázi existuje velké množství a jsou všeobecně známa a používána. Objektová databáze nemá tolik čistých systémů a nejznámější je program SmallTalk. Program Daskalos vyšel vysoko z důvodu, že dotazník byl primárně určen pro žáky informatiky na České zemědělské univerzity, kde se předmět objektová databáze učí pomocí programu Daskalos.



Obrázek 76- Parametr Nahraditelnosti, Zdroj: Vlastní zpracování

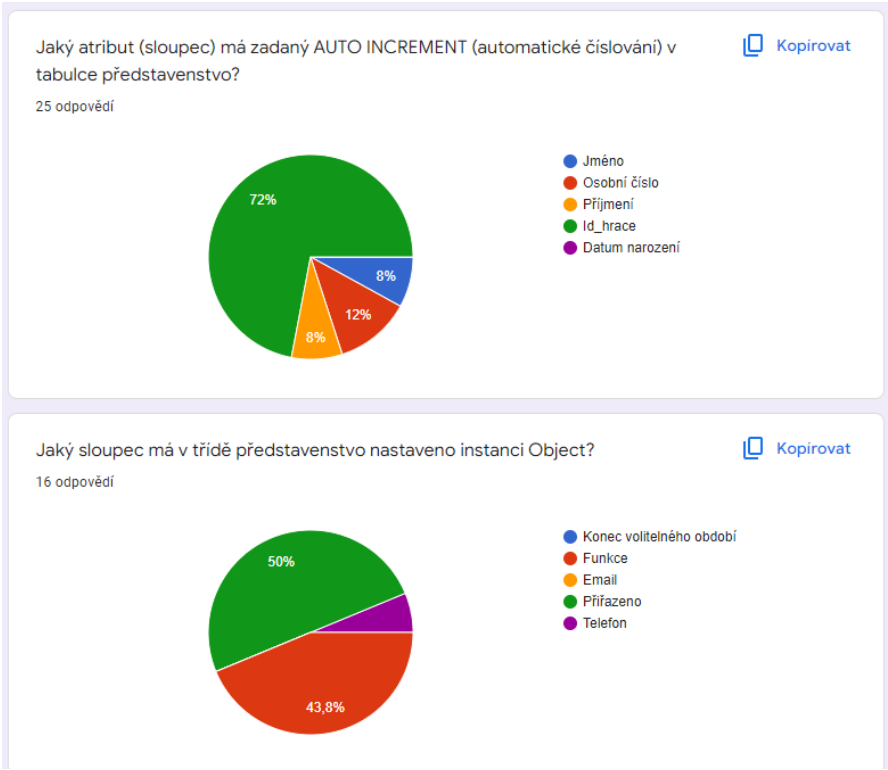
4.5.1.3 Parametr Funkční korektnost

Parametr Funkční korektnost představuje stupeň, jak určitá funkce umožní naplnění cíle a požadavků uživatele. Z objektové a relační databáze měl korespondent zjistit informace o použitých funkcích. V relační databázi byl úkol zjistit, jaké položce ve třídě

Představenstvo je zadána funkce automatického číslování (AUTO INCREMENT). Z objektové databáze měl korespondent zjistit, který sloupec (instance) má nastaven funkci object, která je určena k předání objektu. Správná odpověď v relační databázi je id_hrace. V objektové databázi odpověď přiřazeno.

V relační databázi tento úkol nedělal příliš velký problém a 18 respondentů z 25 odpověděli správně. Úkolem respondenta bylo zapnout relační databází. V PostgreSQL to lze vyhledat v záložce Properties, Containts a přepnout na cizí klíč (Foreign key).

V objektové databázi byl úkol jednoduchý. Na horní liště měl respondent kliknout na záložku Classes a kliknout na třídu Představenstvo. Pod představenstvem se zobrazí instance Představenstvo, kde je vypsáno, že funkci Object má přiřazena instance Přiřazeno.

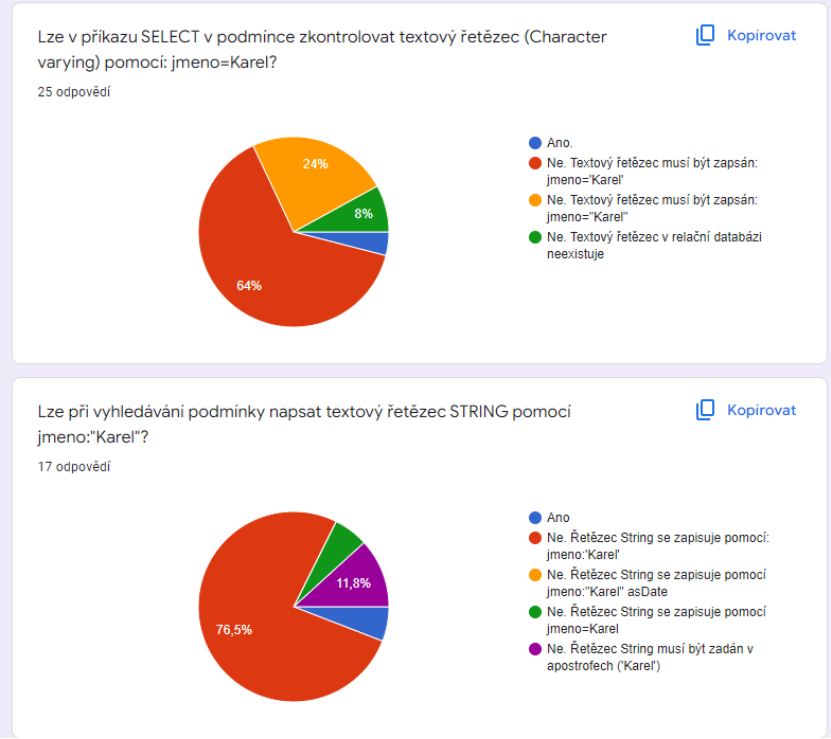


Obrázek 77- Parametr Funkční korektnosti, Zdroj: Vlastní zpracování

4.5.1.4 Parametr Integrity

Parametr integrity ochraňuje špatnou modifikaci dat. V dotazníku měli u obou databází respondenti za úkol vyhodnotit, jak se v dané databázi pracuje s textovým řetězcem (Stringem).

V obou dotazníkách byla správná odpověď stejná. Textový řetězec se v databázích zadává pomocí jednoduchých apostrofů. Vysoké číslo u odpovědi „Ne. Textový řetězec musí být zapsán: jmeno=´“Karel“, může být způsoben, že respondenti si špatně přečetli odpověď a přehlédli uvozovky místo apostrofů.



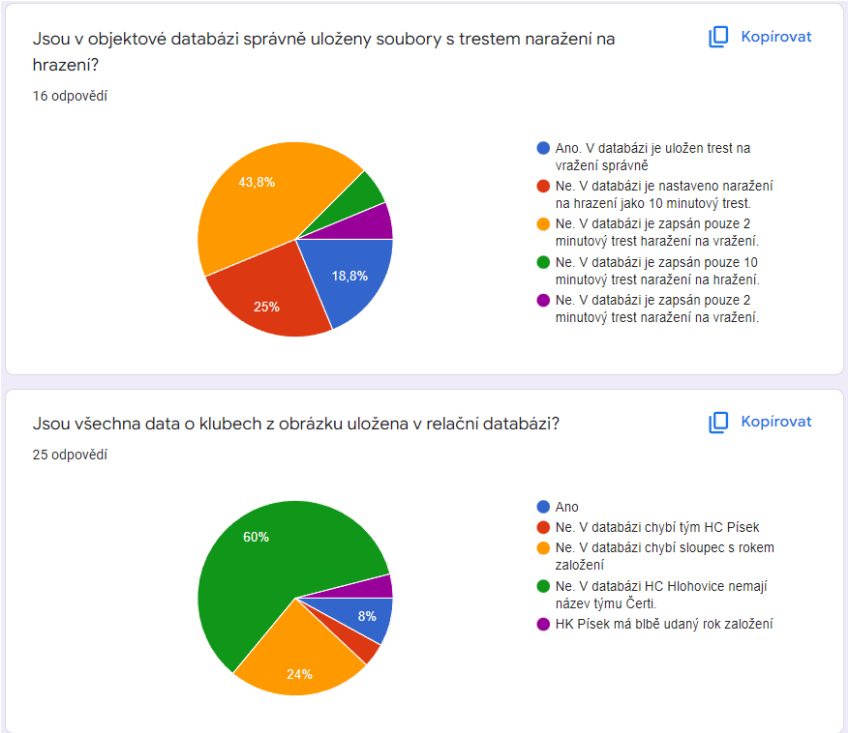
Obrázek 78- Parametr Integrity, Zdroj: Vlastní zpracování

4.5.1.5 Parametr Pravost dat

Pro parametr pravost dat byla testována zadaná data. V obou případech korespondent měl zkontrolovat, zda všechna data z tabulky jsou uložena v databázi.

Výsledky z pravosti dat jsou rozporuplné. V objektové databázi správně odpovědělo pouze 7 respondentů (43,8 %) správně. Vysoké číslo u jiných odpovědí může být způsobeno nepřesnou prací respondentů s textem otázky.

V relační databázi je správná odpověď vyhodnocena 60 %. Respondent měl nalézt, že u týmu HC Hlohovice není nastaven název týmu Čerti.



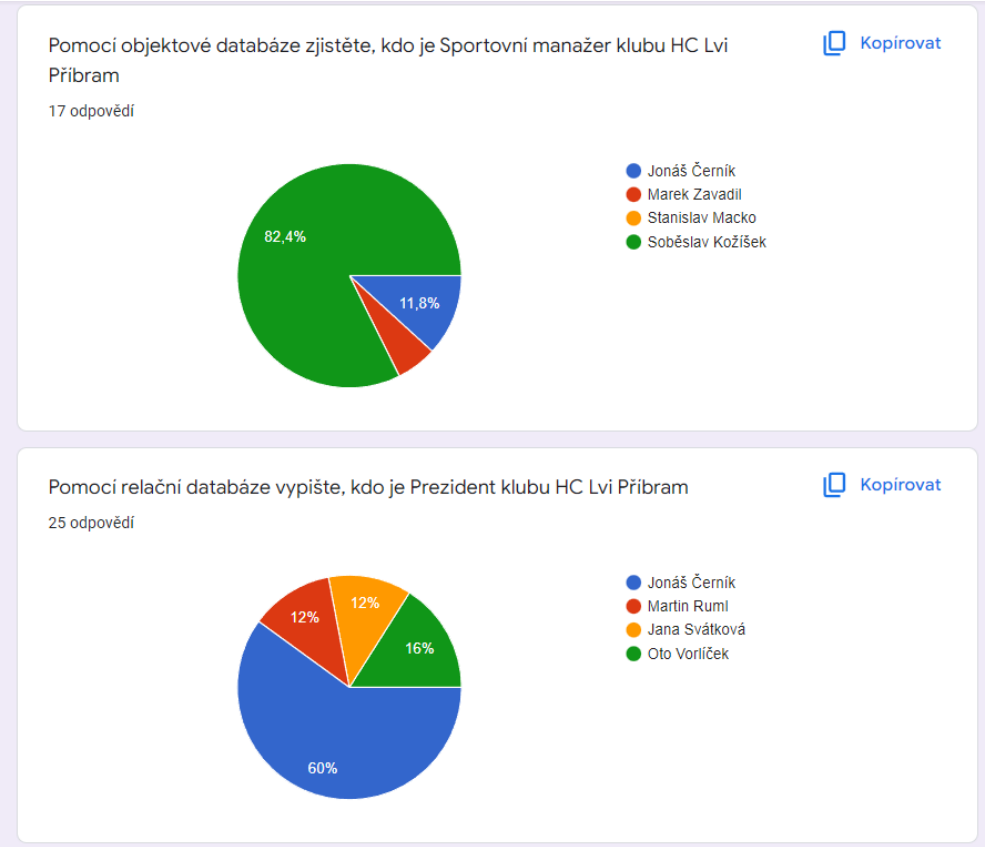
Obrázek 79- Parametr Pravost dat, Zdroj: Vlastní zpracování

4.5.1.6 Parametr Testovatelnost

Pro parametr Testovatelnost byly použity otázky týkající se propojení dvou tabulek. V relační databázi to lze provést pomocí tečkové notace anebo pomocí příkazu JOIN. V objektové databázi to lze provést přímo pomocí scriptu v položce PředstavenstvoSet. Tam lze zkontrolovat, jakou funkci osoba dle otázky vykonává a do kterého klubu patří.

Výsledky průzkumu jsou v tomhle případě více rozdílné. V objektové databázi úspěšnost byla 14 respondentů (82,4 %), což je vysoké procento. Je to nejspíše z důvodu snadné kontroly propojení.

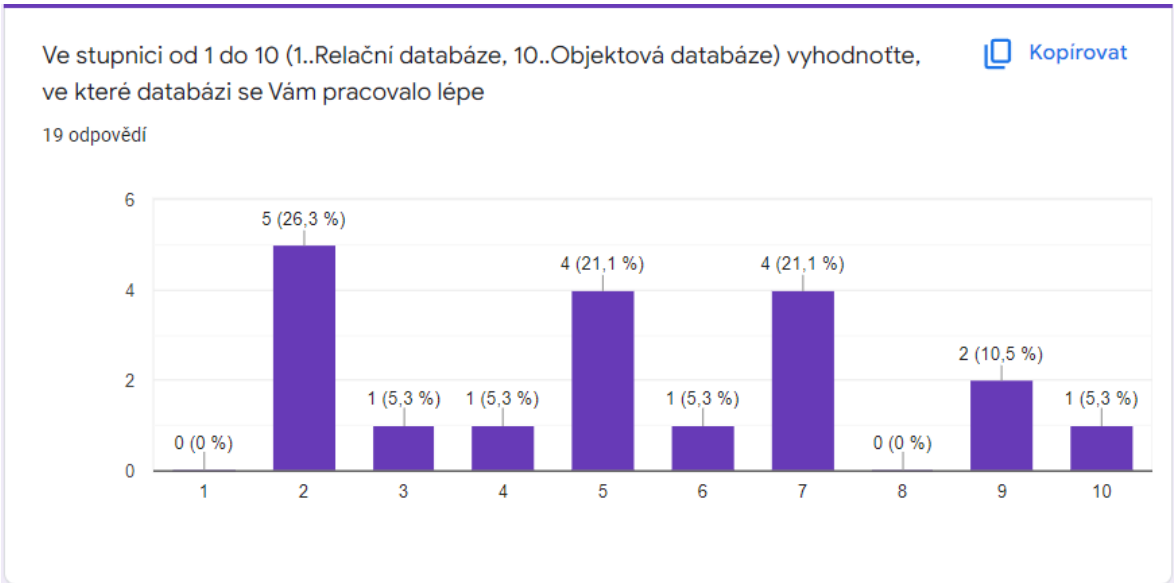
Relační databáze pro propojení je trochu složitější. Respondent musí zjistit jména atributů a identifikační klíče, které pak vypíše do příkazu Select. V relační databázi výsledky dopadly také ve prospěch správné odpovědi- 15 respondentů (60 %). Ostatní odpovědi byly v rozmezí mezi 12 %-16%



Obrázek 80- Parametr Testovatelnost, Zdroj: Vlastní zpracování

4.5.1.7 Parametr Estetičnost

Parametr Estetičnost byl přidán z důvodu zhodnocení respondentů, v jaké databázi se jim lépe pracovalo. Byla zadána bodová škála od 1 do 10. Tento parametr může být trochu zkreslený z důvodu, že se jedná o subjektivní názor každého respondenta. Hodnocení databází dopadlo na podobné úrovni. V relační databázi vyčnívá 5 lidí, kteří odpověděli číslo dvě pro relační databázi. Nejspíše se jedná o respondenty, kteří již déle pracují s relační databází.



Obrázek 81- Parametr Estetičnost, Zdroj: Vlastní zpracování

5 Výsledky a diskuse

V praktické části diplomové práce byl vytvořen dle předchozí analýzy model ve standartu UML. Na jeho vytvoření byla navázána tvorba a naplnění testovacími daty relační a objektové databáze pro hokejový klub. Toto řešení bylo následně porovnáno dle některých parametrů kvality ISO/IEC 25000.

Pro porovnání kvality jednotlivých řešení z pohledu praktické použitelnosti byly zvoleny a popsány instrukce pro založení tabulek, vytvoření a zadání datových atributů a klíčů, vkládání, aktualizaci a mazání dat.

V případě relačních databází byla data vkládána po vytvoření tabulek a nastavení typů jednotlivých atributů, klíčů a funkce autoincrement. Výhoda relační databáze spočívá ve větší možnosti výběru datových typů. Na jeden příkaz lze vložit více dat do jedné tabulky. Další výhodou je jednodušší aktualizace dat. Na rozdíl v objektové databázi je nutné nalézt manuálně řádek, který má být přepsán a přepsat instanci. Mazání tabulky a záznamu je také jednodušší v relační databázi z důvodu nalezení jedinečného identifikačního čísla dle podmínek oproti objektové databázi, kde se musí přesně určit záznam, který se má vymazat.

V objektové databázi lze data uložit bez předchozí úpravy, kdy se vše ukládá do objektu. Data z propojení jednotlivých objektů lze získat přímo pomocí návratové hodnoty a není potřeba další úprava a znalost databáze. Pro vztah M:N se provede vnoření objektu a situace je podobná jako u jednodušších spojení. V objektové databázi není nutné definování jedinečných identifikátorů pro propojení. Stačí pouze vnořit pomocí funkce jeden objekt do cílového objektu. Výpis je pomocí vnořených objektů. Výhoda spočívá v možnosti zobrazení více instancí z jednoho objektu do jedné instance druhého objektu. Nevýhodou objektové databáze je v případě, když vnoření objektů pokračuje přes více tříd. Toto může zpomalit dotaz. Výhodou objektové databáze je možnost dědění, kdy lze vytvořit jedna nadtřída, z které budou následně instance v podtřídě dědit. Problém nastává při mazání instance. V takovém případě je nutné nalézt všechna propojení, kam byla prvotní instance vnořena.

Pro porovnání implementací databázové technologie pro hokejový oddíl z teoretické části byl zvolen dotazník. Ten obsahoval 3 všeobecné úvodní otázky týkající se operačního systému, zařízení a znalosti databází. Další část byla rozdělena na otázky týkající

se jednotlivých databází, aby danou část doplňovali jen osoby, kteří mají zkušenosti s daným typem databází.

Výsledek dotazníku z diplomové práce odpovídá předpokladu autora práce ze zadání o více používané relační databáze 72 %, na rozdíl objektová databáze 55 %. Neznalost ani jedné z uvedených databází byla velice nízká nejspíše z důvodu dotazníkového šetření se zaměřením především na studenty vysoké školy. Výsledek vyššího hodnocení u relační databáze je způsobena, že povědomí o relační databázi je o dost vyšší. Relační databáze se častěji používají hlavně ve spojitosti s webovými aplikacemi

Nejčastěji byly vybrány nejoblíbenější operační systémy Windows 10 a MacOS. Následoval Windows 11, který se nyní řadí jako nejnovější operační systém.

Další otázky byly zvoleny dle parametrů kvality dle ISO/IEC 25000. Jelikož počet korespondentů, kteří vyplnili otázky pro jednotlivé databáze je rozdílný, tak vyhodnocení daného šetření bylo uvedeno v procentech.

V parametru Instalovatelnosti jsou vyhodnoceny obě databáze srovnatelně. Chyba při instalaci se vyskytla při objektové databázi v 17 % a v relační databázi 16 % případů. Většina případů v nenainstalování spočívala ve vedlejších důvodech, kdy uživatel se nechtěl zaregistrovat, čí dlouhá doba instalace.

Parametru Nahraditelnosti se týkal znalosti korespondentů s jednotlivými programy pro databázové modely. Zde byla možnost vybrat i více možností. Pro objektovou databázi byl 100 % znám program SmallTalk, 82 % Daskalos. To potvrdilo předpoklad o všeobecně velice známém programu SmallTalk, vysoké procento Daskalos je nejspíše způsobeno počtem účastníků dotazníku navštěvující vysoké školy. Daskalos také vychází z programu SmallTalk. U relační databáze byl MySQL 72 %, PosgreSQL 64 %, Oracle 56 %. Rozložení u relační databáze je způsobeno, že existuje vice možností.

V parametru Funkční korektnost byl požadavek na nalezení konkrétních údajů z jednotlivých databází. V objektové databázi správně odpovědělo 72 %, v relační 50 %. Rozdíl je nejspíše způsobem nepozorností a špatným vyhodnocením respondentů, protože v relační i objektové databázi jsou tato data snadno přístupná.

Parametr Integrity pro vyhodnocení práce s textem v obou databázích byl vyhodnocen vysokými procenty. V objektové databázi dosahuje 64 % a u relační databáze 76,5 %. Výsledek je srovnatelný z důvodu, že datový typ (řetězec) je u obou databází zadáván

podobně. Větší rozdíl by mohl nastat v případě, kdy by otázka byla položena z hlediska datumu, kde je rozdíl.

V parametru Pravosti dat byla testována zadaná data a jejich kontrola. Byla vybrána data, která souvisela s klubech HC Lvi Příbram. V objektové databázi bylo 43.8% správných odpovědí. Úspěšnost v relační databázi byl 60 %. Nižší úspěšnost u objektové databáze mohla být způsobena nesprávnou kontrolou dat u korespondenta.

Parametr Testovatelnost kontroloval výpis složitějších dat. Zde bylo vysoké procento správných odpovědí 82,4 % u objektové databáze, u relační 60 %. Nižší výsledek nejspíše způsobil složitější postup pro propojení s nalezením odpovídajících klíčů a atributů. Při znalosti objektové databáze bylo nutné pouze zjistit vnořený objekt.

Parametr Estetičnost byl použit pro zhodnocení práce s databází. Zde se jedná o subjektivní názor. Výsledek parametru Estetičnosti vyšla lépe pro relační databázi. Vysvětlením výsledku je nejspíše větší počet uživatelů relační databáze, kteří s ní mají více zkušeností a lépe se orientují v práci s ní. Naopak uživatelé, kteří pracují s databázemi krátce, vzhledem k úkolům označili objektovou databázi.

6 Závěr

Cílem práce bylo pomocí praktického příkladu porovnat relační a objektovou (grafovou) databázovou technologii z pohledu praktické použitelnosti. Práce měla také za úkol seznámit s moderními trendy v databázové technologii a prozkoumat rozdíly mezi nimi.

V teoretické části byl vymezen pojem modelování dat, jeho základní principy a přístupy. Byly popsány jednotlivé datové modely a jejich návrhy. Vzhledem k použití k vytvoření návrhu modelu v praktické části byly podrobněji vysvětleny modely CLASS diagram a Entity Relationship diagram. Druhá část teoretické části seznamovala čtenáře s pojmy databáze a jejími typy. Byly vypsány jednotlivé principy databází a popsána základní funkčnost a práce s nimi. Z jednotlivých typů byl vybrán Relační databázový model a Objektově databázový systém. U obou typů byly popsány jejich prvky, základní principy, relace, příkazy a práce s daty. Vše bylo popsáno s důrazem na rozdíly a trendy mezi těmito databázemi. Dále bylo zde základní seznámení s jazykem Daskalos a s normou ISO/EC 2500 z důvodu použití v teoretické části.

V praktické části byl vytvořen reálný příklad použití relační a objektové databáze pro hokejový klub. Na základě první společné analytické části byl vytvořen CLASS diagram, který představoval návrh pro obě databáze. Dále byla vygenerována data pro naplnění databází. Pro návrh bylo důležité zvolit řešení, které splňovalo požadovaný účel. Pro podporu relační databáze byl vytvořen implementační Entity-Relationship diagram, který byl podpořen vazebními tabulkami. Relační databáze byla vytvořena v databázovém modelu PostgreSQL. Objektová databáze byla vytvořena v programu Daskalos. Objektová databáze vycházela přímo bez dalších úprav z CLASS diagramu.

Porovnáním byla vyhodnocena relační a objektová databáze. Rozdílů u obou databází bylo zjištěno mnoho. Mezi nejvýraznější rozdíl byla nutná identifikace každého záznamu v relační databázi. V objektové databázi nebylo nutné zařadit jednoznačný identifikátor. Propojení bylo jednodušší provést v objektové databázi, kde se jeden objekt vnořil do druhého objektu. Výhodou objektové databáze také byla možnost ve využití dědění a zapouzdření. Relační databáze umožňovala snadnější vkládání dat a pro autora jednodušší sestavování příkazu pro výpis. Výhoda relační databáze také spočívala ve větší možnosti výběru datového typu a jednodušší aktualizaci dat.

Po zpracování relační a objektové databáze byl sestaven dotazník, který představoval otázky dle parametrů z normy ISO/IEC 25000. Dotazníkového šetření se zúčastnilo 36 respondentů. Jednotlivé otázky byly vyhodnoceny pomocí grafů v procentuálním rozmezí z důvodu rozdílného počtu respondentů pro jednotlivé kategorie. Po všeobecných otázkách týkající se systému byl dotazník rozdělen na dvě oblasti. Aby se omezila možnost zkreslení výsledků z důvodu neznalosti, respondent vyplňoval jen část otázek s databází, s kterou již pracoval. Dle dotazníkového šetření dopadl nejlépe pro relační databázi parametr funkční korektnost, nahraditelnost a povědomí o databázi. Naopak pro objektovou databázi dopadl nejlépe parametr testovatelnosti. Autor diplomové práce nezvolil dobrý příklad pro parametr integrity, jelikož zkoumaný textový řetězec se u obou databází zapisuje stejným způsobem. Pro lepší znázornění bylo vhodnější použití pro testování datové typu pro datum.

Pro praktické použití databázových modelů by bylo vhodné doplnit ještě možnosti třídění dat do jednotlivých sezon, naplnit data o historii klubu, případně doplnit fórum, články, ankety. Dotazníkové šetření lze doplnit o další parametry ČSN ISO/IEC 25000. Další možnost pro dotazník je nechat vyplnit dotazník více korespondenty pro větší vzorek výsledků.

7 Seznam použitých zdrojů

1. **Kaluža, Jindřich a Kalužová, Ludmila.** *Modelování dat v informačních systémech.* Praha : Ekopress, s.r.o., 2011. ISBN 978-80-86929-81-1.

2. **Dařena, František.** objekty.html. *akela.mendelu.cz.* [Online] 2004. [Citace: 20. 8 2021.] https://akela.mendelu.cz/~darena/Perl/objekty.html.

3. **1-semanticky-datovy-model.html. *informacni-systemy.studentske.cz.* [Online] [Citace: 1. 9 2021.] https://informacni-systemy.studentske.cz/2009/07/1-semanticky-datovy-model.html.**

4. Chlápek, Ph.D., Ing. Dušan Chlapek, Kučera, Ph.D., Ing Jan a Ph.D. Palovská, RNDr. Helena. *Datové modelování a návrh relační databáze: Sbírka řešených úloh.* 1. místo neznámé : Oeconomica, 2019. str. 168. 978-80-245-2331-6.

5. Peterson, Richard. guru99. *er-diagram-tutorial-dbms.html#3.* [Online] 7. Říjen 2021. [Citace: 20. Říjen 20.] https://www.guru99.com/er-diagram-tutorial-dbms.html#3.

6. odz/tech/213/page05.html. *vovcr.* [Online] [Citace: 15. Prosinec 2021.] https://www.vovcr.cz/odz/tech/213/page05.html.

7. it-slovnik.cz. *entita.* [Online] [Citace: 20. listopad 2021.] https://it-slovnik.cz/pojem/entita.

8. /guide/data-modeling/what-is-entity-relationship-diagram. *https://www.visual-paradigm.com.* [Online] [Citace: 11. listopad 2021.] https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/.

9. Čápka, David. mysql/mysql-tutorial-datove-typy-a-null. *www.itnetwork.cz.* [Online] [Citace: 25. říjen 2021.] https://www.itnetwork.cz/mysql/mysql-tutorial-datove-typy-a-null.

10. /csharp/zaklady/c-sharp-tutorial-typovy-system-podruhe-datove-typy-string. *www.itnetwork.cz.* [Online] [Citace: 25. říjen 2021.] https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-typovy-system-podruhe-datove-typy-string.

11. /sally/psql/erd.php. *sallyx.* [Online] 12. Říjen 2015. [Citace: 1. listopad 2020.] https://www.sallyx.org/sally/psql/erd.php.

12. Zechmeister, ING. Jiří. databazove-systemy-i9edf92d5e371328514bd1c8a321c519c96463.html. *adoc.* [Online] [Citace: 9. listopad

2021.]https://adoc.pub/databazove-systemy i9edf92d5e371328514bd1c8a321c519c96463.html.

13. /pages/er-diagrams. *lucidchart.* [Online] [Citace: 20. 11 2021.] https://www.lucidchart.com/pages/er-diagrams.

14. uml_class_diagram.htm. *.tutorialspoint.* [Online] [Citace: 5. prosinec 2021.] https://www.tutorialspoint.com/uml/uml_class_diagram.htm.

15. 08_diagramy_trid_pouziti.pdf. *ecom.ef.jcu.* [Online] [Citace: 5. leden 2022.] http://ecom.ef.jcu.cz/web2/download/teorie/08_diagramy_trid_pouziti.pdf.

16. Szturcová, Daniela . http://gisak.vsb.cz/. *OOT (Objektově orientované technologie).* [Online] [Citace: 2. leden 2022.] http://gisak.vsb.cz/wikivyuka/images/e/e3/OotxCD.pdf.

17. RADStudio/Sydney/en/Class_Diagram_Definition_(UML_1.5). *docwiki.embarcadero.com.* [Online] [Citace: 10. listopad 2021.] https://docwiki.embarcadero.com/RADStudio/Sydney/en/Class_Diagram_Definition_(UML_1.5).

18. class_diagrams.pdf. *https://cw.fel.cvut.cz.* [Online] [Citace: 5. leden 2022.] https://cw.fel.cvut.cz/old/_media/courses/x33mis/class_diagrams.pdf.

19. Čápka, David. uml-class-diagram-tridni-model. *itnetwork.* [Online] [Citace: 5. leden 2022.] https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model.

20. Čapka, David. navrh/uml/uml-domenovy-model-diagram. *www.itnetwork.cz.* [Online] [Citace: 1. 9 2021.] https://www.itnetwork.cz/navrh/uml/uml-domenovy-model-diagram.

21. Rydval, Slávek. pravidlo-15-asociacni-trida/. *modelovaci-jazyky.cz.* [Online] [Citace: 5. Leden 2022.] https://modelovaci-jazyky.cz/pravidlo-15-asociacni-trida/.

22. Diagram-trid-class-diagram. *docplayer.* [Online] [Citace: 6. Leden 2022.] https://docplayer.cz/416869-Diagram-trid-class-diagram.html.

23. 7.5.0?topic=structure-class-diagrams. *www.ibm.com.* [Online] https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.

24. timeline-of-database-history. *quickbase.* [Online] [Citace: 25. Leden 2022.] https://www.quickbase.com/articles/timeline-of-database-history.

25. Lutkevich, Ben. definition/database. *searchdatamanagement.techtarget.com.* [Online] Září 2021. [Citace: 15. Leden 2022.] https://searchdatamanagement.techtarget.com/definition/database.

26. Hernandez, Michael J. a John, Viescas L. *Myslíme v jazyku SQL tvorba dotazů.* [překl.] Karel Voráček. 1. místo neznámé : Grada Publishing, 2004. str. 380. 80-247-0899-X.

27. databaze. *it-slovnik.* [Online] [Citace: 25. Leden 2022.] https://it-slovnik.cz/pojem/databaze.

28. the-history-of-databases. *thinkautomation.* [Online] [Citace: 15. Leden 2022.] https://www.thinkautomation.com/histories/the-history-of-databases/.

29. hollerith. *columbia.* [Online] 4. Listopad 2021. [Citace: 15. Leden 2022.] http://www.columbia.edu/cu/computinghistory/hollerith.html.

30. Tomáš Jecha, MVP, MCSD. Lehky-uvod-teorie-databazi. *dotnetportal.* [Online] 12. Prosinec 2009. [Citace: 26. Leden 2022.] https://www.dotnetportal.cz/clanek/60/Lehky-uvod-teorie-databazi.

31. what-is-a-hierarchical-database-model.html. *netinbag.* [Online] [Citace: 25. Leden 2022.] https://www.netinbag.com/cs/internet/what-is-a-hierarchical-database-model.html.

32. understanding-the-network-database-model. *mariadb.* [Online] [Citace: 26. Leden 2022.] https://mariadb.com/kb/en/understanding-the-network-database-model/.

33. Peterson, Richard. introduction-to-database-sql. *guru99.* [Online] 11. Listopad 2021. [Citace: 26. Leden 2022.] https://www.guru99.com/introduction-to-database-sql.html.

34. 4.-Základy-relačních-databází-jejich-využití-v-programování-webu.pdf. *kme.vse.* [Online] 16. Srpen 208. [Citace: 2. Únor 2022.] https://kme.vse.cz/wp-content/uploads/page/534/4.-Z%C3%A1klady-rela%C4%8Dn%C3%ADch-datab%C3%A1z%C3%AD-jejich-vyu%C5%BEit%C3%AD-v-programov%C3%A1n%C3%AD-webu.pdf.

35. 9733715-sql-keys. *cs.education-wiki.* [Online] [Citace: 2. Únor 2022.] https://cs.education-wiki.com/9733715-sql-keys.

36. difference-between-primary. *cs.gadget-info.* [Online] 2019. [Citace: 2. Únor 2022.] https://cs.gadget-info.com/difference-between-primary.

37. Peart, Matt. define-alternate-keys-reference-records. *docs.microsoft.* [Online] 7. Září 2021. [Citace: 2. Únor 2022.] https://docs.microsoft.com/cs-cz/powerapps/maker/data-platform/define-alternate-keys-reference-records.

38. Qiu, Marry. database-normalization-description. *docs.microsoft.* [Online] 1. Únor 2022. [Citace: 6. Únor 2022.] https://docs.microsoft.com/cs-cz/office/troubleshoot/access/database-normalization-description.

39. Kulhan, Jakub a Lehocký, Zdeněk. 2008071900-normalizace-relacnich-databazi/. *programjte.* [Online] 23. Leden 2008. [Citace: 4. Únor 2022.] http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/.

40. normal-forms-in-dbms/. *geeksforgeeks.* [Online] [Citace: 10. Únor 2022.] https://www.geeksforgeeks.org/normal-forms-in-dbms/.

41. Normalizace_datab%C3%A1ze#%C4%8Ctvrt%C3%A1_norm%C3%A1ln%C3%AD_forma_(4NF). *czwiki.* [Online] [Citace: 10. Únor 2022.] https://czwiki.cz/Lexikon/Normalizace_datab%C3%A1ze#%C4%8Ctvrt%C3%A1_norm%C3%A1ln%C3%AD_forma_(4NF).

42. Kalčev, Ing. Petr. UvodDoDatabazi.pdf. *people.fsv.cvut.* [Online] [Citace: 15. Únor 2022.] http://people.fsv.cvut.cz/~dlaskpet/Help/UvodDoDatabazi.pdf.

43. page20.html. *vovcr.* [Online] [Citace: 5. Únor 2022.] https://www.vovcr.cz/odz/tech/393/page20.html.

44. what-is-data-redundancy.html. *netinbag.* [Online] https://www.netinbag.com/cs/internet/what-is-data-redundancy.html.

45. data-manipulation-language-dml. *techopedia.* [Online] 13. Říjen 2014. [Citace: 5. Únor 2020.] https://www.techopedia.com/definition/1179/data-manipulation-language-dml.

46. data-definition-language-ddl. .*techopedia.* [Online] 21. Září 2020. [Citace: 2. Únor 2022.] https://www.techopedia.com/definition/1175/data-definition-language-ddl.

47. sql-ddl-dql-dml-dcl-tcl-commands/. *geeksforgeeks.* [Online] 30. Září 2021. [Citace: 15. Únor 2022.] https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/.

48. HE_SQL2_GRANT.htm. *sql602.sourceforge.* [Online] [Citace: 15. Únor 2022.] http://sql602.sourceforge.net/helpdir-cs/html/HE_SQL2_GRANT.htm.

49. sql-select. *javatpoint.* [Online] [Citace: 15. Únor 2022.] https://www.javatpoint.com/sql-select.

50. joins.php. *techonthenet.* [Online] [Citace: 16. Únor 2022.] https://www.techonthenet.com/sql/joins.php.

51. sql_update.asp. *.w3schools.* [Online] [Citace: 20. Únor 2022.] https://www.w3schools.com/sql/sql_update.asp.

52. HE_SQL2_INSERT.htm. *sql602.sourceforge.* [Online] [Citace: 20. Únor 2022.] http://sql602.sourceforge.net/helpdir-cs/html/HE_SQL2_INSERT.htm.

53. sql_delete.asp. *w3schools.* [Online] [Citace: 20. Únor 2022.] https://www.w3schools.com/sql/sql_delete.asp.

54. Zedníček, Ing. Jan. sql-create-table-zalozeni-tabulky/. *biportal.* [Online] 30. Březen 2017. [Citace: 15. Únor 2022.] https://biportal.cz/sql-create-table-zalozeni-tabulky/.

55. sql_autoincrement.asp. *w3schools.* [Online] [Citace: 15. Únor 2022.] https://www.w3schools.com/sql/sql_autoincrement.asp.

56. sql_check.asp. *w3schools.* [Online] [Citace: 15. Únor 2022.] https://www.w3schools.com/sql/sql_check.asp.

57. sql_alter.asp. *w3schools.* [Online] [Citace: 15. Únor 2022.] https://www.w3schools.com/sql/sql_alter.asp.

58. sql_ref_drop_table.asp. *w3schools.* [Online] [Citace: 15. Únor 2022.] https://www.w3schools.com/sql/sql_ref_drop_table.asp.

59. sql-rename-table. *javatpoint.* [Online] [Citace: 15. Únor 2022.] https://www.javatpoint.com/sql-rename-table.

60. sql_comments.asp. *w3schools.* [Online] [Citace: 15. Únor 2022.] https://www.w3schools.com/sql/sql_comments.asp.

61. what-are-object-oriented-databases/. *thecustomizewindows.* [Online] 16. Únor 2021. [Citace: 24. Únor 2022.] https://thecustomizewindows.com/2021/02/what-are-object-oriented-databases/.

62. Dancuk Milica. object-oriented-database. *phoenixnap.* [Online] [Citace: 24. Únor 2022.] https://phoenixnap.com/kb/object-oriented-database.

63. 3551156-Datove-modelovani-vojtech-merunka.html. *docplayer.* [Online] [Citace: 24. Únor 2022.] http://docplayer.cz/3551156-Datove-modelovani-vojtech-merunka.html.

64. guide/uml-unified-modeling-language/uml-class-diagram-tutorial/. *visual-paradigm.* [Online] [Citace: 1. listopad 2021.] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/.

65. sql_drop_table.asp. *w3schools.* [Online] [Citace: 15. Únor 2022.] https://www.w3schools.com/sql/sql_drop_table.asp.

66. Merunka, Ph.D., Ing. Vojtěch, Pergl, Ing. Robert a Pícka, Ing. Marek. *OBJEKTOVĚ ORIENTOVANÝ PŘÍSTUP.* místo neznámé : ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE, 2005.

67. object-oriented-database. *phoenixnap.* [Online] [Citace: 24. Únor 2022.] https://phoenixnap.com/kb/object-oriented-database.