

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informatika

Návrh databáze pro střední školu

Bakalářská práce

Student : **Martin Schoř**
Vedoucí bakalářské práce : **RNDr. Helena Palovská, Ph.D.**
Oponent bakalářské práce : **Ing. Dušan Chlapek, Ph.D.**

ROK: 2012

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 9. 5. 2012

.....

podpis

Poděkování

Tímto bych chtěl poděkovat všem učitelům, kteří mi poskytli informace o střední škole potřebné k vytvoření této práce, jmenovitě pak hlavně panu Vladislavu Hampejsovi. Dále bych rád poděkoval paní RNDr. Heleně Palovské, Ph.D. za poskytnuté konzultace a vedení bakalářské práce. Samozřejmě bych chtěl také poděkovat celé své rodině za podporu při tvorbě této práce a při studiu.

Abstrakt

Cílem této práce bylo vytvořit alternativní, jednodušší databázi bez zbytečných, nevyužitých funkcí použitelnou pro aplikaci určenou k evidenci žáků a učitelů a správu rozvrhů střední školy. Databáze by také měla být snadno rozšiřitelná o další funkce.

Postupoval jsem od analýzy střední školy přes konceptuální a fyzický návrh databáze až k nastavení práv jednotlivých uživatelů. Databázi jsem také na malém vzorku dat vyzkoušel v Oracle Database Express Edition 11g Release 2.

Předpokládaným přínosem práce by mělo být zpřehlednění a zjednodušení stávajícího stavu zaznamenávání dat na konzultovaných středních školách pro učitele. Přínosem by pak měla být práce hlavně pro žáky škol, kterým by umožnila nahlížet na záznamy o jejich osobě včetně známek, k čemuž v současné době přístup nemají.

Práce je strukturována hlavně kolem praktického návrhu databáze s občasným doplněním o teorii k tématu.

Klíčová slova:

návrh databáze, konceptuální model, střední škola

Abstract

The goal of this bachelor thesis is to create alternative and simpler database without unused functions, which can be used by application designed to keep a records of students and teachers and to create and manage timetable of high school. The created database should also be easily extendable.

I started by analyzing high school, then created conceptual and physical model of the database and in the end described access rights of users of the database. I've also used a small data sample to test the database in Oracle Database Express Edition 11g Release 2.

The expected contribution of this thesis for teachers should be more transparent and simpler database than the one which is currently used by the consulted high schools. The created database should mainly be a contribution for students, who currently have no access at all to the school database.

The thesis is mainly focused on the database design and its practical realization with theoretical parts where relevant.

Key words:

conceptual model, high school, database desgin

Obsah

1. Úvod.....	1
1.1 Popis tématu.....	1
1.2 Přínos práce	1
1.3 Způsob vypracování	1
1.4 Předpoklady a omezení práce	2
2. Tvorba databáze	3
2.1 Logický návrh dle Michaela J. Hernandeze	3
2.2 Použitý postup	4
3. Střední škola a současný stav	6
3.1 Střední škola.....	6
3.2 Současný stav	7
3.3 Alternativní systémy a práce	8
4. Analýza funkcí.....	9
5. Návrh databáze.....	11
5.1 Konceptuální model.....	11
5.2 Fyzický model.....	21
5.3 Integritní omezení.....	21
6. Případy užití a přístupová práva	26
6.1 Případy užití	26
6.2 Přístupová práva	29
7. Závěr.....	32
8. Terminologický slovník.....	34
9. Zdroje	35
10. Přílohy.....	37
10.1 Tabulky	37
10.2 Konceptuální model.....	38
10.3 Fyzický model.....	39
10.4 SQL skript.....	40
10.5 Pohledy	50
10.6 CHECK a Trigger.....	51

1. Úvod

1.1 Popis tématu

Tématem bakalářské práce je vytvořit návrh databáze pro konkrétní aplikaci. Budu navrhovat databázovou strukturu schopnou podporovat aplikaci pro evidenci žáků, učitelů a tvorbu a prohlížení rozvrhů na střední škole. Téma jsem si zvolil z důvodu zájmu o databáze a především jejich správu, avšak tvorba databází mne také zaujala a ve spojení s dobrou možností konzultace v rámci problematiky fungování střední školy jsem se rozhodl pro dané téma.

Ačkoliv konzultované školy již databázové systémy mají, shodují se jejich uživatelé, že obsahují množství nevyužívaných funkcí.

Cílem práce tedy je navrhnout funkční databázi pro střední školu, která podpoří požadované funkce a nebude obsahovat funkce nevyužívané. Informace o fungování střední školy a jejích požadavcích na záznam dat jsem získával ve spolupráci s několika učiteli dvou děčínských středních škol s rozdílným zaměřením a zkušenostmi v oboru.

1.2 Přínos práce

Tato práce si klade za cíl vytvořit alternativní, jednodušší databázi bez zbytečných, nevyužitých funkcí použitelnou pro obě konzultované střední školy s možností případného snadného doplnění o další v současné době nevyužívané funkce.

Pro učitele by mohla přinést zprehlednění a zjednodušení stávajícího stavu.

Přínosem by pak měla být hlavně pro žáky škol, kterým by umožnila nahlížet na záznamy o jejich osobě včetně známek, k čemuž v současné době přístup nemají.

1.3 Způsob vypracování

Asi každý má nějakou představu, jak funguje střední škola, stručně tedy popíšu základní specifika střední školy a rozdíly dvou středních škol, od nichž jsem získával informace. Pro dosažení cíle práce použiji různé metody získané studiem předmětů na vysoké škole a z použité literatury[1] rozdělené do několika etap. V první etapě na základě vlastních zkušeností ze střední školy a na základě rozhovorů s učiteli analyzuji požadované funkce, navrhnu jednotlivé entity, atributy a vztahy mezi entitami. V druhé etapě na základě poznámek vytvořím konceptuální model databáze, ve kterém stanovím datové typy a omezení jednotlivých atributů. Ve třetí etapě vygeneruji z konceptuálního modelu fyzický model, v němž provedu v případě potřeby nutné úpravy. Ve čtvrté etapě popíši případy užití a přístupová práva uživatelů. Na konec vytvořím pomocí generovaného a doplněného SQL kódu fyzickou strukturu databáze.

1.4 Předpoklady a omezení práce

Návrh databáze bude primárně určen pro aplikaci určenou k evidenci žáků a učitelů a správu rozvrhů. Nejedná se tedy o aplikaci pokrývající veškerou činnost střední školy. Správa nepedagogických pracovníků není v databázi zahrnuta.

2. Tvorba databáze

Tvorba databáze se obecně skládá ze tří fází[1]:

- **Logický návrh** je „první fáze zahrnující určení a definování tabulek a jejich polí, stanovení primárních a cizích klíčů, zřízení vztahů mezi tabulkami a zjištění a zřízení různých úrovní integrity dat“[1].
- **Fyzická implementace**, kdy je předchozí návrh zrealizován za pomoci vhodných nástrojů.
- **Vývoj aplikace**, kdy se pro databázi vytvoří aplikace umožňující uživatelům práci s daty v databázi uloženými.

Tato práce se zabývá téměř výhradně fází logického návrhu a databázi jsem fyzicky implementoval pouze pro účely otestování funkčnosti návrhu.

Logický návrh databáze není těžký proces a s trochou informací by ho zvládl asi každý. Bez správného, naučeného postupu a důkladného uvážení by se však velmi snadno mohlo stát, že takto navržená databáze bude obsahovat duplicitní či nekonzistentní data a vyhledávání v ní bude obtížné a bude vracet nepřesné údaje. „Nepřesné údaje jsou pravděpodobně nejhorším důsledkem špatného návrhu databáze“[1]. Je tedy vhodné naučit se a použít správnou metodologii a těmto problémům se vyhnout. Metodologií a přístupů k návrhu databáze je mnoho a já osobně jsem tvorbu této práce založil na znalostech a zkušenostech z kurzů o databázích na Vysoké škole ekonomické, kde jsem získal hlavně praktické poznatky a z knihy Michaela J. Hernandezze[1] z které jsem čerpal hlavně teoretický podklad.

2.1 Logický návrh dle Michaela J. Hernandezze

Kniha Michaela J. Hernandezze[1] popisuje proces návrhu databáze v sedmi hlavních bodech. Důsledné dodržení těchto bodů není těžké, avšak může být časově náročné, přesto je však kladen důraz na důsledné provedení a dokončení celého procesu jelikož jen tak je zajištěna maximální spolehlivost a integrita dat. Hledání zkratk a zjednodušení v procesu pak vede pouze k nutnosti předělání databáze a další časové a popřípadě i jiné ztrátě.

Sedm bodů popsaného procesu návrhu je následujících.

Určení formulace úkolu a cílů úkolu

Je první fází procesu návrhu databáze. Formulace úkolu určuje a definuje účel databáze. Cíle úkolu definují obecné úkony, které budou s daty v databázi provádět uživatelé. Formulace úkolů a jejich cílů dohromady slouží jako vodítko během návrhu databáze.

Analýza existující databáze

V případě že nějaká databáze v organizaci již existuje, přistoupí se v druhé fázi k její analýze. Takováto databáze bývá buď zděděná (databáze funkčně používaná již delší dobu) nebo papírová (volné propojení papírových záznamů).

Analýza se dělí na dvě části a to samotnou analýzu databáze, způsob získávání, uchovávání a prezentace dat a na pohovory s uživateli zjišťující, jak je databáze běžně používána.

Vytváření datových struktur

V další fázi se na základě definovaných cílů úkoly a analýzy dat vytvoří entity, z kterých se stanou tabulky a přiřadí se jim odpovídající pole, která reprezentují různé vlastnosti dané entity. Z polí se jedno či více vybere a určí jako primární klíč. Pro každé pole v tabulce by také měla být vytvořena jeho specifikace.

Určení a zřízení vztahů mezi tabulkami

Čtvrtá fáze je určení vztahů mezi tabulkami. Tyto vztahy se nejlépe určí dalšími rozhovory s managementem a uživateli. Pomocí primárních a cizích klíčů a případně vazebních tabulek se pro tyto vztahy zřídí logická spojení, kterým se určí typ a stupeň účasti, který bude v některých případech zjevný již z podstaty ukládaných dat, jindy se určí na základě business pravidel.

Určení a definování business pravidel

V páté fázi se opět na základě rozhovorů určí omezení založená na způsobu, jak organizace prezentuje a zpracovává data. Obecná omezení vztahující se spíše na vztahy mezi tabulkami lépe zjistíme od managementu, specifická omezení vztahující se spíše k polím nám lépe sdělí samotní uživatelé. Součástí tvorby omezení je také tvorba validačních tabulek.

Určení a definování pohledů

Pro realizaci šesté fáze opět provedeme rozhovory a zjistíme, kdo v organizaci používá jaká data pro svou práci, a definujeme pohledy, které zajistí přístup právě k těmto datům.

Kontrola integrity dat

Sedmou a poslední fází procesu návrhu je kontrola integrity dat. Konkrétně se jedná o kontrolu, že tabulky splňují požadavky správného návrhu, kontrolu specifikací polí, kontrolu platnosti vztahů a kontrola business pravidel.

2.2 Použitý postup

Skloubením vlastní představy o procesu návrhu a metod popsanych v knize Michaela J. Hernandez[1] jsem začal tvořit tuto databázi. Původně jsem více dal na své postupy, ale záhy se ukázalo, že nejsou moc efektivní a mnohem více jsem tedy začal používat postupy popsané v knize. Ve výsledku je tedy databáze vytvořená převážně dle procesu z literatury jen s odchylkami v případech, kdy popsané metody byly nepoužitelné, neproveditelné či zbytečné z důvodů omezení práce. Například analýza databáze mi nebyla ani v jedné ze škol umožněna a tato fáze tak byla provedena jen pomocí rozhovorů s uživateli. Ačkoliv kniha obsahuje pouze logický návrh databáze a nikoliv fyzickou implementaci, která je v práci

částečné také obsažena, je většina postupů v této práci založena právě na této knize.

3. Střední škola a současný stav

3.1 Střední škola

Následuje po ukončení 9. stupně základní školy a poskytuje odborné či úplné všeobecné vzdělání. Studium střední školy trvá 3 nebo 4 roky (výjimku tvoří osmiletá gymnázia, kdy jde stále o 4 roky studia střední školy, ale předchází jim 4 roky dokončení základního vzdělání) podle svého zaměření a výsledné kvalifikace studentů a je ukončeno maturitní zkouškou, nebo výučním listem. Školský zákoník rozlišuje následující typy středních škol:

- gymnázium
- střední odborná škola
- střední odborné učiliště

Organizačně se střední škola člení na třídy. Střední vzdělávání se uskutečňuje ve formě denní, večerní dálkové, distanční a kombinované. Vzdělání dosažené ve všech těchto formách je rovnocenné. Vyučovací hodina trvá 45 minut, vyučovací hodina odborného výcviku a odborné praxe trvá 60 minut. Školní docházka na střední škole je povinná, avšak lze ji plnit i formou individuálního vzdělávání. Cílem středního vzdělání je rozvíjet vědomosti, dovednosti, schopnosti, postoje a hodnoty získané v základním vzdělávání, které jsou důležité pro osobní rozvoj jedince.[4]

Mnou konzultované střední školy, jmenovitě Gymnázium Děčín (gymnázium) a Střední škola lodní dopravy a technických řemesel v Děčíně (střední odborná škola, dále SOŠ) jsou poměrně typickými zástupci středních škol svých typů, obě školy nabízejí pouze denní studijní obory s výjimkou nástavbového maturitního oboru na SOŠ. Gymnázium nabízí pouze obor všeobecného vzdělávání, a to v podobě čtyřleté s maturitou a osmileté s maturitou, kdy jde o kombinaci druhého stupně základní školy s plynulým přechodem na školu střední. Systém výuky v prvních čtyřech ročnících osmiletého gymnázia (tedy prakticky základní školy) se téměř neliší od systému výuky v dalších letech, databáze tak nemá problém obsáhnout i tuto část výuky. SOŠ nabízí velké množství specializovaných tříletých oborů zakončených závěrečnou zkouškou a nástavbový dvouletý obor s maturitou.

Žáci jsou na obou školách standardně rozděleni do tříd kdy každá třída má jednoho třídního učitele. Třídy spolu zůstávají po celou dobu studia a většinu, někdy i všechny předměty mají stejné. Rozdíly většinou tvoří výuka jazyků, kdy se jedna část třídy učí jiný jazyk než druhá část. V posledních dvou letech gymnázia se také rozvrhy jednotlivých žáků stejné třídy liší dle vybraných volitelných předmětů.

3.2 Současný stav

Bakaláři

Obě dotazované střední školy v současné době používají systém Bakaláři. Systém bakaláři se skládá z několika modulů, z nichž některé jsou samostatně použitelné a jiné jsou závislé na ostatních modulech. Oficiální popis systému bakaláři je následující: „Program Bakaláři řeší evidenci žáků (školní matrika) a zaměstnanců, klasifikaci (zápis známek, tisk vysvědčení a třídních výkazů, grafické zpracování prospěchu), přípravu úvazků, sestavení rozvrhu hodin, plánování akcí školy, suplování. Další moduly programu Bakaláři slouží pro přijímací řízení resp. zápis do prvního ročníku, inventarizaci majetku, rozpočet školy, půjčování knih a učebnic, rozpis maturitních zkoušek, tvorbu tematických plánů atd.“[11] Tento oficiální popis nastiňuje moduly tvořící systém bakaláři, kterými jsou:

Evidence žáků a zaměstnanců – zpracovává osobní údaje žáků, jejich rodičů a zaměstnanců školy. Součástí modulu je i klasifikace žáků a tisk vysvědčení. Modul má několik podmodulů: Třídní kniha, Webové aplikace, Přijímací zkoušky, zápis do 1. ročníku ZŠ, Grafické zpracování klasifikace a Rozpis maturit.

Tento modul používají obě školy, protože se jedná v podstatě o základní modul celého systému a podporuje hlavní funkci školy, tj. výuka žáků. Klasifikaci lze evidovat průběžně i pololetně. V současné době obě školy do systému zaznamenávají pouze pololetní klasifikaci a průběžnou si řeší učitelé dle svých preferencí. Funkce podmodulů odpovídají jejich názvům. Třídní kniha je elektronickou verzí klasické třídní knihy, jelikož obě školy používají papírovou třídní knihu, tento modul není využíván. Modul webové aplikace je zaměřen na zpřístupnění vybraných dat z ostatních modulů především rodičům žáků, ale umožňuje i přístup žákům samotným. Modul je jen omezeně využíván na gymnáziu (je přístupný jen rodičům) a není využíván na SOŠ. Přijímací zkoušky, zápis do 1. ročníku ZŠ je modul zaměřený na evidenci a zpracování dat z přijímacích zkoušek na střední školu, variantou je verze pro zápis do prvních ročníků ZŠ. Tento modul ani jedna škola nevyužívá. Grafické zpracování klasifikace je aplikace pro tvorbu grafů a zobrazení časového vývoje klasifikace. Aplikaci ani jedna škola nevyužívá. Rozpis maturit je aplikace pro sestavení rozvrhu maturit. Aplikaci ani jedna škola nevyužívá.

Knihovna, Inventarizace, Rozpočet školy, Plán akcí školy a Tematické plány – zaměřují se na vedení a řízení školy. Ani jedna škola tyto moduly nevyužívá a řeší tyto oblasti jinými způsoby nebo vůbec (SOŠ nemá vlastní knihovnu).

Rozvrh a Suplování – aplikace pro tvorbu a tisk rozvrhů a suplování. Aplikace modulu rozvrh asistuje při manuální tvorbě rozvrhů kontrolou kolizí a návrhy změn, umožňuje také automatické nebo poloautomatické generování rozvrhů. Aplikaci využívají obě školy. Aplikace modulu suplování asistuje při tvorbě

suplování nabízením volných učitelů, místností či změnami hodin a spojením skupin. Aplikaci využívají obě školy.

3.3 Alternativní systémy a práce

Mimo současného stavu na konzultovaných středních školách jsem se také pokusil vyhledat alternativní informační systémy a práce na téma databází pro střední školu. Za použití internetových vyhledavačů jsem našel článek[12] zabývající se nabídkou a kritérii pro výběr informačního systému pro školu. Článek v závěru zmiňuje některé nejrozšířenější informační systémy, které bych doplnil o další mnou nalezené systémy a to AMOS-IS a Etřídnice které taky patří mezi významné a v článku nejsou zmíněny. Po prostudování propagačních materiálů těchto systémů mohu konstatovat, že z většiny nabízejí rámcově stejnou funkcionalitu jako již zmíněný systém bakaláři až na tři výjimky, systémy iškola a Škola OnLine, které jsou pouze online systémy bez klientských aplikací a systém aSc Rozvrhy, který se soustředí pouze na rozvrhy a suplování, jedná se také o jediný mezinárodně využívaný systém ze systémů v článku a mnou jmenovaných.

Při hledání prací na téma databáze pro střední školu jsem využil veřejné části registru systému Theses.cz. Na dané téma jsem však nenalezl žádné práce, rozšířil jsem tedy vyhledávání na informační systémy pro střední školy a již jsem našel několik prací. Nalezené práce byly vesměs podobné a zabývaly se analýzou požadavků, návrhem informačního systému a do různé míry i jeho implementací. Diplomová práce Tomáše Kormaňáka se zabývá vývojem „informačního systému pro evidenci studijních výsledků studentů základních a středních škol a online zpřístupnění těchto výsledků rodičům“[13]. Stejně jako má práce si klade za cíl zjednodušení stávajícího stavu informačních systémů na středních školách, obsahuje však tvorbu celého informačního systému včetně webové aplikace, prostřednictvím které je systém přístupný školám. Diplomová práce Stanislava Juřicy se obdobně zabývá vytvořením „informačního a prospěchového systému pro základní a střední školy“[14]. Práce se také zabývá vývojem celého informačního systému, klade však hlavní důraz na open source řešení a dostupnost zdarma. Bakalářská práce Bc. Maroše Džoganíka jejímž cílem je „analýza, návrh a čiastočná implementácia informačného systému pre základné a stredné školy“[15] se opět zabývá vývojem celého informačního systému, zakládá však práci především na poznatcích ze základní školy a implementuje jen vzorovou část systému. Bakalářská práce Miroslava Brabence[16] se zabývá studií informačního systému pro středoškolské učitele orientovaného především na funkci elektronické třídní knihy. Práce si klade za cíl seznámení s problematikou vývoje informačních systémů.

4. Analýza funkcí

Před samotným návrhem databáze je třeba určit funkce aplikace, kterou má databáze podporovat. Abych tyto funkce určil, nejdříve na základě rozhovorů s uživateli a jimi provedeného předvedení stávajícího systému uvedu funkce stávajícího systému a jejich využití na obou školách. Jak již bylo uvedeno v kapitole 3.2, obě školy používají systém bakaláři, funkce uvedené v tabulce 1 tedy odpovídají modulům tohoto systému.

#	Funkce	Gymnázium	SOŠ
1	evidence žáků a zaměstnanců	využívá	využívá
2	zpracování klasifikace	využívá	využívá
3	tvorba a tisk vysvědčení	využívá	využívá
4	grafické zpracování klasifikace	nevyužívá	nevyužívá
5	třídní kniha – záznam jednotlivých hodin a absence žáků na nich	nevyužívá	nevyužívá
6	tvorba a správa tematických plánů, rozpočtu a plánů akcí školy	nevyužívá	nevyužívá
7	funkce knihovny a inventární funkce	nevyužívá	nevyužívá
8	tvorba a tisk rozvrhů a suplování	využívá	využívá
9	rozpis maturit a správa dat k přijímacím zkouškám	nevyužívá	nevyužívá
10	komunikace s rodiči a žáky	využívá částečně	nevyužívá

Tabulka 1, analýza funkcí

Jak je vidět z tabulky, obě školy využívají systém bakaláři jen omezeně a hlavně využívají funkce a aplikace přímo související s výukou a ostatní funkce potřebné pro běh školy jsou řešeny jiným způsobem.

Funkcemi využívanými na obou školách tedy jsou: evidence žáků a zaměstnanců, zpracování klasifikace, tvorba a tisk vysvědčení a tvorba a tisk rozvrhů a suplování. Tyto funkce tedy bude podporovat i mnou vytvářená databáze (ačkoliv v rámci omezení nebudou evidováni všichni zaměstnanci ale pouze učitelé). Jelikož mnou vytvářená databáze není určena pro modulovou aplikaci, budou některé funkce pozměněny, aby lépe odpovídaly struktuře databáze. Ačkoliv systém bakaláři podporuje komunikaci (respektive sdělování informací) žákům, jeho webová aplikace pro tuto funkci určená je spíše orientována na rodiče. Aplikace mé databáze by se měla orientovat přímo na žáka a databáze bude podporovat tedy funkci sdělování informace žákům.

Po změně některých stávajících využívaných funkcí a po přeorientování komunikace na žáka je seznam databází podporovaných funkcí následující:

- evidence žáků, jejich zákonných zástupců, absencí a výchovných opatření
- evidence učitelů
- zpracování pololetní klasifikace, tvorba a tisk vysvědčení

- zpracování mimopololetních zkoušek (maturity, reparaáty)
- tvorba a tisk rozvrhů
- sdělování informací o studiu žákům

5. Návrh databáze

5.1 Konceptuální model

Na základě funkcí, které má databáze podporovat, stanovených v kapitole 4 vytvořím entity zajišťující tyto funkce, výjimkou je funkce sdělování informací o studiu žákům, která bude řešena pouze pohledem, jak bude popsáno v kapitole 6.2. V této kapitole budou popsány jednotlivé entity a jejich atributy. Namapování entit na funkce je k nalezení v tabulce 2 v příloze 10.1.

Entity jsou označovány množným číslem a velkými písmeny, atributy číslem jednotným, kdy první písmeno slova je velké a mezery jsou nahrazeny podtržítky.[1] Konceptuální model je k nalezení v příloze 10.2.

Entity, atributy, datové typy

Zde popíši entity jako takové, jejich atributy, datové typy atributů a jejich případná omezení.

ZACI

Entita slouží k evidenci osobních údajů žáků, kteří studují či studovali na škole.

ID_Zaka	- datový typ INTEGER - primární klíč - unikátní číselná identifikace žáka v rámci školy - minimální hodnota je 1000 z důvodu unikátnosti čísel žáků a učitelů
Jmeno	- datový typ VARIABLE CHARACTERS s maximální délkou 20 znaků - povinný atribut zaznamenávající křestní jméno žáka
Prijmeni	- datový typ VARIABLE CHARACTERS s maximální délkou 25 znaků - povinný atribut zaznamenávající příjmení žáka
Ulice	- datový typ VARIABLE CHARACTERS s maximální délkou 35 znaků - povinný atribut zaznamenávající ulici bydliště žáka
Mesto	- datový typ VARIABLE CHARACTERS s maximální délkou 35 znaků - povinný atribut zaznamenávající město bydliště žáka
Cislo_Popisne	- datový typ INTEGER - povinný atribut zaznamenávající číslo popisné bydliště žáka
PSC	- datový typ VARIABLE CHARACTERS s maximální délkou 6 znaků - povinný atribut zaznamenávající poštovní směrovací číslo bydliště žáka
Rodne_Cislo	- datový typ VARIABLE CHARACTERS s maximální délkou 11 znaků - povinný atribut zaznamenávající rodné číslo žáka

Telefon	- datový typ CHARACTERS s délkou 12 znaků - zaznamenává telefonní číslo žáka
Datum_Narozeni	- datový typ DATE - povinný atribut zaznamenávající datum narození žáka
Cislo_Tr_Vykazu	- datový typ INTEGER - povinný atribut zaznamenávající číslo žáka v třídním výkazu
Trida	- datový typ VARIABLE CHARACTERS s maximální délkou 5 znaků - povinný datový atribut určující třídu, do které žák v současnosti náleží
Cizi_Jazyk	- datový typ VARIABLE CHARACTERS s maximální délkou 15 znaků - povinný datový atribut zaznamenávající cizí jazyk, který žák studoval na základní škole
Nastup	- datový typ DATE - povinný datový atribut zaznamenávající datum nástupu žáka do školy
Ukončení	- datový typ DATE - při ukončení studia se zde zaznamená datum odchodu žáka ze školy

PRERUSENI_STUDIA

Entita slouží k evidenci jednotlivých přerušení studia žáků. Entita je závislá na entitě ZACI.

Prer_Od	- datový typ DATE - tvoří složený primární klíč - společně s cizím klíčem ID_Zaka jednoznačně identifikují každé přerušení studia - zaznamenává začátek přerušení studia
Prer_Do	- datový typ DATE - povinný atribut zaznamenávající konec přerušení studia
Prer_Duvod	- datový typ VARIABLE CHARACTERS s maximální délkou 50 znaků - povinný atribut zaznamenávající důvod přerušení studia

OBORY

Entita eviduje seznam na škole vyučovaných oborů.

Zkratka_Obor	- datový typ VARIABLE CHARACTERS s maximální délkou 5 znaků - primární klíč - jednoznačně označuje každý obor
--------------	---

Nazev_Obor - datový typ VARIABLE CHARACTERS s maximální délkou 50 znaků
- povinný atribut zaznamenávající plný název oboru

VYCHOVNA_OPATRENI

Entita eviduje výchovná opatření udělená žákům. Entita je závislá na entitě ZACI a entitě TYPY_VYCH_OPATRENI.

Datum_Opatreni - datový typ DATE
- tvoří složený primární klíč
- společně s cizími klíči ID_Zaka a ID_Typ_Vych_Op jednoznačně identifikují každé udělené výchovné opatření
- zaznamenává datum udělení výchovného opatření

Duvod_Udeleni - datový typ TEXT
- povinný atribut se zdůvodněním udělení výchovného opatření

TYPY_VYCH_OPATRENI

Entita eviduje jednotlivé typy možných výchovných opatření (např. napomenutí třídního učitele, pochvala ředitele školy)

ID_Typ_Vych_Op - datový typ INTEGER
- primární klíč
- jednoznačně identifikuje každý typ výchovného opatření

Typ_Opatreni - datový typ VARIABLE CHARACTERS s maximální délkou 30 znaků
- povinný atribut zaznamenávající celý název typu výchovného opatření

ZAKONI_ZASTUPCI

Entita eviduje zákonné zástupce žáků školy. Atributy Jmeno, Prijmeni, Ulice, Cislo_Popisne, PSC, Mesto a Telefon jsou shodné s atributy entity ZACI, jen se vztahují k zákonným zástupcům žáků.

ID_Zak_Zastupce - datový typ INTEGER
- primární klíč
- jednoznačně identifikuje každého zákonného zástupce

SKUPINY

Entita eviduje jednotlivé skupiny žáků potřebné k tvorbě rozvrhů rozdělené podle společného vyučování. V této skupině jsou jak klasické třídy, tak skupiny v daných třídách (např. žáci 1A učící se angličtinu). U skupin je zaznamenán školní rok, pro který byly platné pro možnost archivního dohledání zařazení žáků do tříd.

ID_Skupiny	- datový typ INTEGER - primární klíč - jednoznačně identifikuje každou skupinu
Nazev_Skupiny	- datový typ VARIABLE CHARACTERS s maximální délkou 30 znaků - povinný atribut zaznamenávající název skupiny
Skolni_Rok	- datový typ CHARACTERS s délkou 9 znaků - povinný datový atribut zaznamenávající, v kterém školním roce byla daná skupina aktuální

ABSENCE

Entita eviduje souhrn absencí žáků za jednotlivá pololetí jednotlivých ročníků. Entita je závislá na entitě ZACI.

A_Skolni_Rok	- datový typ CHARACTERS s délkou 9 znaků - tvoří složený primární klíč - zaznamenává školní rok, v kterém byla absence zaznamenána
A_Pololeti	- datový typ BYTE - tvoří složený primární klíč - zaznamenává pololetí, kterého se daná absence týká - minimální hodnota je 1, maximální 2
A_Rocnik	- datový typ INTEGER - povinný atribut zaznamenávající ročník studia, v kterém byla absence zaznamenána - minimální hodnota je 1, maximální 8
Zmeskano	- datový typ INTEGER - povinný atribut zaznamenávající kolik hodin žák za celé pololetí zameškal
Neomluveno	- datový typ INTEGER - povinný atribut zaznamenávající kolik hodin má žák za pololetí neomluvených

ZNAMKY

Entita evidující pololetní známky žáků pro vysvědčení. Entita je závislá na entitě ZACI a entitě PREDMETY.

Znamka	- datový typ CHARACTERS s délkou jednoho znaku - povinný atribut zaznamenávající udělenou známku, případnou neklasifikaci či uvolnění z daného předmětu - povolené hodnoty jsou '1', '2', '3', '4', '5', 'N' a 'U'
--------	--

Rocnik	- datový typ INTEGER - povinný atribut zaznamenávající v jakém ročníku byl žák, když mu byla známka udělena - minimální hodnota je 1, maximální 8
Pololeti	- datový typ BYTE - tvoří složený primární klíč - zaznamenává pololetí, v kterém byla známka udělena - minimální hodnota je 1, maximální 2
Datum_Znamka	- datový typ DATE - povinný atribut zaznamenávající, kdy byla známka udělena
Z_Skolni_Rok	- datový typ CHARACTERS s délkou 9 znaků - tvoří složený primární klíč - zaznamenává školní rok, v kterém byla známka udělena

DUVODY_UKONCENI

Pro lepší přehlednost a použití entita eviduje důvody ukončení studia.

ID_Ukonceni	- datový typ INTEGER - primární klíč - jednoznačně identifikuje každý důvod ukončení studia
Duvod_Ukonceni	- datový typ VARIABLE CHARACTERS s maximální délkou 50 znaků - povinný atribut s přesným popisem důvodu ukončení studia

KURZY

Smyslem této entity je evidovat výsledky tvorby rozvrhů, tedy jednotlivé vyučované hodiny během týdne v aktuálním pololetí. Entita je závislá na entitách SKUPINY a HODINY.

Den	- datový typ VARIABLE CHARACTERS s maximální délkou 7 znaků - tvoří složený primární klíč - zaznamenává den, v kterém je daná hodina vyučována - povolené hodnoty jsou „PONDELI“, „UTERY“, „STREDA“, „CTVRTEK“ a „PATEK“
Tyden	- datový typ VARIABLE CHARACTERS s maximální délkou 5 znaků - povinný datový atribut značící zda se předmět vyučuje v sudém nebo lichém týdnu nebo v obou - povolené hodnoty jsou „LICHY“, „SUDY“ a „OBA“

HODINY

Entita je seznamem hodin pro výuku s jejich začátkem a koncem.

- Hodina - datový typ INTEGER
- primární klíč
 - jednoznačně identifikuje každou hodinu
- Zacatek - datový typ TIME
- povinný atribut značící začátek vyučované hodiny
- Konec - datový typ TIME
- povinný atribut značící konec vyučované hodiny

PREDMETY

Entita eviduje veškeré předměty, které se na škole vyučují nebo vyučovaly. Eviduje i předměty, které se přímo nevyučují, ale jsou předmětem zkoušek.

- Zkratka_Pred - datový typ VARIABLE CHARACTERS s maximální délkou 5 znaků
- primární klíč
 - jednoznačně identifikuje každý vyučovaný předmět
- Nazev_Pred - datový typ VARIABLE CHARACTERS s maximální délkou 20 znaků
- povinný atribut zaznamenávající plný název předmětu
- Typ_Pred - datový typ CHARACTERS s délkou jednoho znaku
- povinný atribut značící typ předmětu, zda je povinný, volitelný nebo zda se jedná o cizí jazyk
 - povolené hodnoty jsou „P“, „V“ a „J“

UCITELE

Entita slouží k evidenci učitelů, kteří na škole vyučují. Atributy Jmeno, Prijmeni, Ulice, Mesto, Cislo_Popisne, PSC, Rodne_Cislo, Telefon, Nastup a Ukonceni jsou stejné jako u entity ZACI, jen se vztahují k učitelům.

- ID_Ucitele - datový typ INTEGER
- primární klíč
 - unikátní číselná identifikace učitele v rámci školy
 - maximální hodnota je 999 z důvodu unikátnosti čísel žáků a učitelů
- Telefon_Pr - datový typ CHARACTERS s délkou 12 znaků
- zaznamenává pracovní telefonní číslo učitele
- Praxe_Obor - datový typ INTEGER
- povinný atribut zaznamenávající počet let odpracovaných učitelem v oboru
- Praxe_Mimo - datový typ INTEGER

- povinný atribut zaznamenávající počet let odpracovaných učitelem mimo obor

MISTA

Entita eviduje místa, na kterých může probíhat výuka nebo zkoušky.

- Zkr_Mista - datový typ VARIABLE CHARACTERS s maximální délkou 5 znaků
- primární klíč
 - zkratka plného názvu místa, jednoznačně identifikuje každé místo
- Místo - datový typ VARIABLE CHARACTERS s maximální délkou 50 znaků
- povinný atribut zaznamenávající plný název místa
- Typ_Mista - datový typ CHARACTERS s délkou jednoho znaku
- povinný atribut značící zda se místo nachází v areálu školy či mimo
 - povolené hodnoty jsou „S“ a „M“

ZKOUSKY

Entita eviduje zkoušky, které žáci na škole vykonali v době, kdy již byli žáky, neeviduje tedy zkoušky přijímací. Jedná se i o jednotlivé zkoušky, které mohou dohromady tvořit jednu větší zkoušku (např. jednotlivé části maturity). Entita je závislá na entitě ZACI a entitě PREDMETY.

- Datum_Zkouska - datový typ DATE
- tvoří složený primární klíč
 - zaznamenává datum, kdy byla zkouška konána
- Vysledek - datový typ CHARACTERS s délkou jednoho znaku
- povinný atribut zaznamenávající známku udělenou ze zkoušky ne zda se žák nedostavil
 - povolené hodnoty jsou „1“, „2“, „3“, „4“, „5“ a „N“
- Forma - datový typ CHARACTERS s délkou 2 znaků
- povinný atribut značící zda zkouška byla provedena písemně, ústně či prakticky
 - povolené hodnoty jsou „PI“, „US“ a „PR“

TYPY_ZKOUSEK

Entita eviduje typy zkoušek, které mohou na škole probíhat.

- ID_Typu_Zk - datový typ INTEGER
- primární klíč
 - jednoznačně identifikuje každý typ zkoušky
- Typ_Zk - datový typ VARIABLE CHARACTERS s maximální délkou 30 znaků
- povinný atribut s plným názvem typu zkoušky

Vztahy mezi entitami

Popis jednotlivých vztahů mezi entitami. V modelu jsou vztahy pojmenovány bez diakritiky a začínají malými písmeny.

Zastupuje

Vztah mezi entitami ZACI a ZAKONI_ZASTUPCI. Zákonní zástupci zastupují jednotlivé žáky. Každý žák musí mít právě jednoho zákonného zástupce. Každý zákonný zástupce musí zastupovat aspoň jednoho žáka, může jich však zastupovat i více.

Uskutečňuje

Vztah mezi entitami ZACI a ABSENCE. Žáci uskutečňují absence. Žák může mít více absencí, ale nemusí mít žádnou, což nastává v případě prvního pololetí studia. Každá absence musí být přiřazena právě jednomu žákovi.

Ukončují studium z

Vztah mezi entitami ZACI a DUVODY_UKONCENI. Žáci ukončují studium z důvodů ukončení. Každý žák po ukončení studia má přiřazen právě jeden důvod ukončení, dokud studuje, důvod ukončení nemá. Z jednoho důvodu může ukončit studium více žáků ale také nikdo.

Náleží k

Vztah mezi entitami ZACI a PRERUSENI_STUDIA. Přerušení studia náleží k žákům. Každé přerušení studia musí náležet právě k jednomu žákovi. Žák může mít více přerušení studia, ale nemusí mít žádné.

Je vyučován v

Vztah mezi entitami ZACI a OBORY. Žák je vyučován v oboru. Každý žák musí být vyučován právě v jednom oboru. V daném oboru nemusí v určitém čase být vyučován nikdo, ve většině oborů je však vyučováno více žáků.

Jsou udělena

Vztah mezi entitami ZACI a VYCHOVNA_OPATRENI. Výchovná opatření jsou udělena žákům. Každé výchovné opatření musí být přiděleno právě jednomu žákovi. Žákovi může být uděleno více výchovných opatření, ale nemusí mu být uděleno žádné.

Je variantou

Vztah mezi entitami VYCHOVNA_OPATRENI a TYPY_VYCH_OPATRENI. Výchovná opatření jsou variantou typu výchovného opatření. Každé výchovné opatření musí být právě jednoho typu. Jednoho typu může být více opatření, ale také žádné.

Je udělena

Vztah mezi entitami ZACI a ZNAMKY. Znamka je udělena žákovi. Každému žákovi může být uděleno několik známek, ale nemusí mu být udělena žádná. Každá známka musí být udělena právě jednomu žákovi.

Je zařazen do

Vztah mezi entitami ZACI a SKUPINY. Žáci jsou zařazeni do skupin. Žáci mohou být zařazeni do více skupin, ale nemusí být zařazeni do žádné. V každé skupině musí být alespoň jeden žák, ale může jich být i více.

Je udělena z

Vztah mezi entitami ZNAMKY a PREDMETY. Znamky jsou udělovány z předmětů. Každá známka musí být udělena právě z jednoho předmětu. Z každého předmětu může být uděleno několik známek, ale nemusí být udělena žádná.

Má být učena

Vztah mezi entitami SKUPINY a PREDMETY. Skupina má být učena předměty. Jedná se o vztah primárně určený pro podporu tvorby rozvrhů. Několika skupinám je přiřazeno několik předmětů, které mají mít v rozvrhu. Skupiny, které již nejsou vyučovány, nemají přiděleny žádné předměty a předměty, které již nejsou vyučovány, nejsou přiděleny žádným skupinám.

Je vyučován pro

Vztah mezi entitami SKUPINY a KURZY. Kurz je vyučován pro skupinu. Každý kurz musí být vyučován právě pro jednu skupinu. Každá skupina může být vyučována v několika kurzech, ale nemusí být aktuálně vyučována v žádném.

Se vyučuje z

Vztah mezi entitami KURZY a PEDMETY. Kurz se vyučuje z předmětu. Každý kurz musí být vyučován právě z jednoho předmětu. Z jednoho předmětu může být vyučováno několik kurzů, ale také žádný.

Vyučuje

Vztah mezi entitami UCITELE a KURZY. Učitel vyučuje kurz. Každý učitel může vyučovat několik kurzů, ale nemusí vyučovat žádný. Každý kurz musí být vyučován právě jedním učitelem.

Se vyučuje v

Vztah mezi entitami KURZY a MISTA. Kurz se vyučuje v místě. Každý kurz se musí vyučovat právě v jednom místě. Na každém místě může být vyučováno několik kurzů, ale také žádný.

Může vyučovat

Vztah mezi entitami UCITELE a PREDMETY. Učitel může vyučovat předměty. Jedná se o vztah primárně určený pro podporu tvorby rozvrhů. Několika učitelům je přiděleno několik předmětů, které mohou vyučovat. Učitel však nemusí

vyučovat žádný předmět a předmět nemusí být aktuálně vyučován žádným učitelem.

Se koná v

Vztah mezi entitami MISTA a ZKOUSKY. Zkouška se koná v místě. Každá zkouška se musí konat právě na jednom místě. Na jednom místě se může konat více zkoušek ale nemusí se tam konat žádná.

Je konána z

Vztah mezi entitami ZKOUSKY a PREDMETY. Zkouška je konána z předmětu. Každá zkouška musí být konána právě z jednoho předmětu. Z jednoho předmětu může být konáno několik zkoušek, ale nemusí být konána žádná.

Vykonává

Vztah mezi entitami ZACI a ZKOUSKY. Žák vykonává zkoušku. Žák může vykonat několik zkoušek, ale také nemusí vykonat žádnou. Každá zkouška musí být vykonána právě jedním žákem.

Je typem

Vztah mezi entitami ZKOUSKY a TYPY_ZKOUSEK. Zkouška je typem typu zkoušky. Každé zkoušce musí být přiřazen právě jeden typ. Z jednoho typu může být konáno více zkoušek, ale také nemusí být konána žádná.

Čas výuky

Vztah mezi entitami KURZY a HODINY. Hodina zaznamenává čas výuky kurzu. Každý kurz se vyučuje právě v jednu hodinu. V jedné hodině může být vyučováno několik kurzů ale také žádný.

Je třídní

Vztah mezi entitami UCITELE a SKUPINY. Učitel je třídním učitelem skupiny (třídy). Každý učitel může být třídním učitelem pouze jedné skupiny (třídy), ale nemusí být třídním učitelem žádné. Každá skupina může mít pouze jednoho třídního učitele, ale nemusí mít žádného.

5.2 Fyzický model

Fyzický model jsem jednoduše vygeneroval z konceptuálního modelu. Po vygenerování modelu jsem přejmenoval cizí klíče a upravil integritní omezení, kde bylo třeba. Přejmenoval jsem také vazební tabulky a reference, protože vygenerované názvy mi nepřišly vhodné.

Fyzický model je k nalezení v příloze 10.3.

5.3 Integritní omezení

Primární klíče

Primární klíč je pole nebo kombinace polí, která jednoznačně identifikuje každý záznam v databázové tabulce. Každé pole, které je součástí primárního klíče, musí být řádně vyplněno. Každá tabulka má definovaný právě jeden primární klíč.[5]

Cizí klíče

Cizí klíč je integritní omezení, které zajišťuje propojení jednoho či více sloupců tabulky s jedním nebo více sloupci jiné tabulky (cizí).[6] Primární klíč jedné tabulky se pak stává cizím klíčem tabulky jiné.

Referenční integrita

Referenční integrita vzniká pomocí cizích klíčů a zajišťuje integritu dat mezi propojenými tabulkami tak, že při smazání záznamu v rodičovské tabulce vyvolá určitou akci. Standardní nastavení akce na RESTRICT (pokud na rodičovský záznam existuje odkaz nelze ho smazat) není vhodné pro všechny reference. Je tedy vhodné reference upravit a nastavit akci dle potřeby na CASCADE (záznamy odkazující na smazaný záznam se smažou také). Další možné akce jsou SET DEFAULT (při smazání rodičovského záznamu se odkazy v dceřiné tabulce nastaví na výchozí hodnotu) a SET NULL (při smazání rodičovského záznamu se odkazy v dceřiné tabulce nastaví na hodnotu NULL), tyto akce jsem však nepoužil.[6]

Záznamy v tabulkách žáci, učitelé a skupiny se za normálních okolností nemazou. Pokud by však z nějakého důvodu bylo nutné záznam v některé z těchto tabulek vymazat, stávají se ostatní záznamy na tyto záznamy odkazující bezvýznamné a je vhodné je také smazat, referenční integrita s tím počítá a je tedy nastavena následovně:

FK_ABSENCE_ZACI

Při vymazání žáka jsou smazány i jeho absence.

FK_KURZY_SKUPINY

Nelze vymazat skupinu, pokud je pro ni vyučován kurz.

FK_KURZY_MISTA

Nelze vymazat místo, pokud na daném místě probíhá výuka nějakého kurzu.

FK_KURZY_PREDMETY

Nelze vymazat předmět, pokud z něj probíhá výuka.

FK_KURZY_UCITELE

Nelze vymazat učitele, pokud právě vyučuje nějaký kurz. Pokud je nutné učitele smazat, je třeba nejdříve přiřadit kurz jinému učiteli.

FK_PRERUSEN_ZACI

Při smazání žáka se smažou i jeho přerušení studia.

FK_PRIRAZENI_SKUPINY

Při smazání skupiny se zruší i přiřazení předmětů této skupině.

FK_PRIRAZENI_PREDMETY

Nelze smazat předmět, pokud je přiřazen nějaké skupině.

FK_VYCHOVNA_TYPY

Nelze smazat typ výchovného opatření, pokud existuje udělené výchovné opatření tohoto typu.

FK_VYCHOVNA_ZACI

Při smazání žáka se smažou i jemu udělená výchovná opatření.

FK_VYUKA_UCITELE

Při smazání učitele se smaže i seznam předmětů, které může vyučovat.

FK_VYUKA_PREDMETY

Při smazání předmětu se smažou i přiřazení tohoto předmětu jako možné k výuce u jednotlivých učitelů. Jelikož však předmět nelze smazat, pokud na něj jakákoliv jiná tabulka v databázi odkazuje, nastane tato situace zcela výjimečně.

FK_ZACI_OBORY

Obor nelze smazat, pokud jsou nebo byli v daném oboru vyučováni žáci.

FK_ZACI_DUVODY

Důvod ukončení nelze smazat, pokud z daného důvodu nějaký žák ukončil studium.

FK_ZACI_ZAKONI

Zákonného zástupce nelze smazat, pokud zastupuje nějakého žáka.

FK_ZARAZENI_ZACI

Při smazání žáka dojde i k zrušení jeho zařazení do skupin.

FK_ZARAZENI_SKUPINY

Při smazání skupiny dojde i k zrušení zařazení žáků do této skupiny.

FK_ZKOUSKY_PREDMETY

Předmět nelze smazat, pokud z něj proběhla nějaká zkouška.

FK_ZKOUSKY_TYPY

Typ zkoušky nelze smazat, pokud proběhla nějaká zkouška tohoto typu.

FK_ZKOUSKY_MISTA

Místo nelze smazat, pokud na daném místě proběhla nějaká zkouška.

FK_ZKOUSKY_ZACI

Při smazání žáka se smažou i záznamy o všech jeho zkouškách.

FK_ZNAMKY_ZACI

Při smazání žáka se smažou i známky, které mu byly uděleny.

FK_ZNAMKY_PREDMETY

Předmět nelze smazat, pokud z něj byla udělena nějaká známka.

FK_KURZY_HODINY

Hodinu nelze smazat, pokud je v ní vyučován nějaký kurz.

FK_SKUPINY_UCITELE

Nelze smazat učitele, pokud je třídním skupiny (třídy).

FK_UCITELE_SKUPINY

Při smazání skupiny se zruší záznam o ní u jejího třídního učitele.

Omezení a business pravidla

Jedná se o omezení polí v případech, kdy by určité hodnoty nedávaly logicky smysl (např. konec musí mít pozdější datum než začátek) nebo kdy jsou hodnoty omezeny na základě pravidel fungování organizace.

Tato omezení zajišťují SQL příkaz CHECK generovaný automaticky dle nastavení omezení v konceptuálním modelu anebo ručně vytvořené TRIGGERy.

CHECK

Omezení se vztahují na tabulku nebo všechny výskyty atributu, který je uveden v názvu CHECKu.

CHECK ABSENCE

Omezení zajišťující, že žák nebude mít více neomluvených absencí než zameškaných hodin.

CHECK UCITEL

Omezení zajišťující, že odchod učitele bude vždy později než jeho nástup na školu.

CHECK ZACI

Omezení zajišťující, že odchod žáka bude vždy později než jeho nástup na školu.

CHECK HODINY

Omezení zajišťující, že konec hodiny bude vždy později než její začátek.

CHECK ID_Ucitele

Omezení zajišťující, že ID učitele nebude větší než 999. Omezení existuje z důvodu, aby bylo možné mít unikátní čísla pro žáky i učitele. Omezení nepředpokládá se, že by škola během existence databáze zaměstnala více jak 999 učitelů. Pokud by k tomu mělo dojít bylo by třeba toto omezení stanovit vyšší.

CHECK Zkratka_Pred

Omezení zajišťující, že písmena ve zkratce předmětu budou vždy velká.

CHECK Tyden

Omezení zajišťující, že týden výuky bude vždy SUDY (sudý), LICHY (lichý) nebo NULL (předmět je vyučován každý týden).

CHECK Typ_Mista

Omezení zajišťující, že typ místa bude buď S (ve škole) nebo M (mimo školu)

CHECK Zkratka_Obor

Omezení zajišťující, že písmena ve zkratce oboru budou vždy velká.

CHECK Typ_Pred

Omezení zajišťující, že typ předmětu bude vždy P (povinný), V (volitelný) nebo J (cizí jazyk).

CHECK PRERUSENI_STUDIA

Omezení zajišťující, že konec přerušení studia bude vždy později než začátek přerušení studia.

CHECK PSC

Omezení zajišťující, že písmena v PSC budou vždy velká. Omezení existuje pro podporu poštovních směrovacích čísel zemí, které používají v poštovním směrovacím číslu písmena.

CHECK Cislo_Tr_Vykazu

Omezení zajišťující, že číslo žáka v třídním výkazu bude vždy větší než nula.

CHECK Vysledek

Omezení zajišťující, že výsledek zkoušky bude vždy 1, 2, 3, 4, 5, nebo N (nedostavil se).

CHECK Forma

Omezení zajišťující, že forma zkoušky bude vždy PI (písemně), US (ústně) nebo PR (praktická).

CHECK ID_Zaka

Omezení zajišťující, že ID žáka bude vždy minimálně 1000. Omezení existuje z důvodu, aby bylo možné mít unikátní čísla pro žáky i učitele. Omezení nepředpokládá se, že by škola během existence databáze zaměstnala více jak 999 učitelů. Pokud by k tomu mělo dojít bylo by třeba toto omezení stanovit vyšší.

CHECK Znamka

Omezení zajišťující, že udělená známka bude vždy 1, 2, 3, 4, 5, N (neklasifikován) nebo U (uvolněn).

CHECK Rocnik

Omezení zajišťující, že ročník bude vždy číslo mezi 1 a 8. Omezení na 8 je pro podporu víceletých gymnázií, pro klasické střední školy by bylo možné omezení změnit na 1 až 4.

CHECK Pololeti

Omezení zajišťující, že pololetí bude vždy 1 nebo 2, tedy první nebo druhé.

CHECK vložení důvodu ukončení

Omezení zajišťující, že při vložení ukončení studia bude určen i důvod ukončení studia. Tento CHECK je vytvořen ručně.

TRIGGER

Ověření začátku přerušení

Omezení zajišťující, že přerušení studia žáka bude začínat pouze v době studia žáka.

Ověření ukončení přerušení

Omezení zajišťující, že přerušení studia žáka bude končit pouze v době studia žáka.

Ověření data výchovného opatření

Omezení zajišťující, že datum výchovného opatření bude pouze v době studia žáka.

Ověření data známky

Omezení zajišťující, že datum udělení známky bude pouze v době studia žáka.

Ověření data zkoušky

Omezení zajišťující, že datum konání zkoušky bude pouze v době studia žáka.

6. Případy užití a přístupová práva

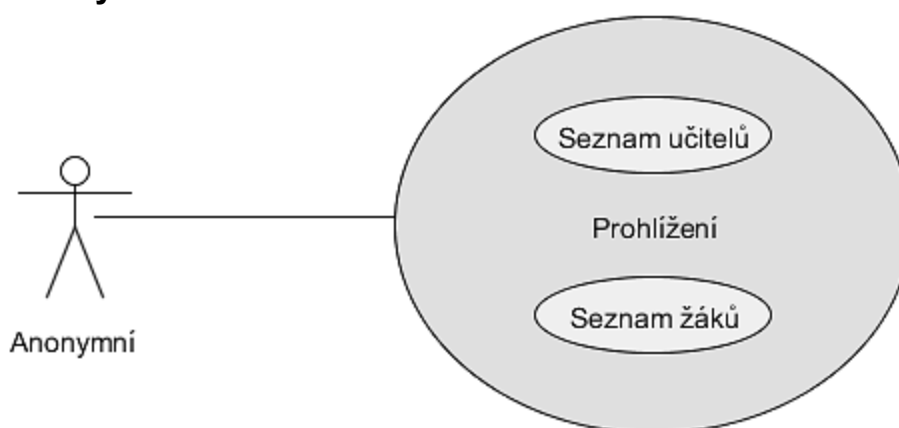
Případy užití a přístupová práva uživatelů popíší společně v jedné kapitole. Případy užití popisují, jak bude moci daný uživatel databázi používat. Přístupová práva jsou konkrétní povolené akce pro určité tabulky nebo pohledy.

Základní uživatelé databáze jsou: Anonymní, Žák, Učitel, Administrativní pracovník, Ředitel školy a Externí kontrola. Tito uživatelé jsou i rolemi, které budou v databázi vytvořeny. Přístupová práva budou přidělena právě těmto rolím a uživatelské účty budou následně do těchto rolí přiřazovány. Toto usnadní správu přístupů pro správce databáze, kdy nemusí každému jednotlivému uživateli (kterých budou v případě žáků stovky) přidělovat zvlášť práva. Jednotliví uživatelé se budou identifikovat heslem. [8]

6.1 Případy užití

Případy užití jednotlivých uživatelů znázorním use case diagramy s popisem.

Anonymní



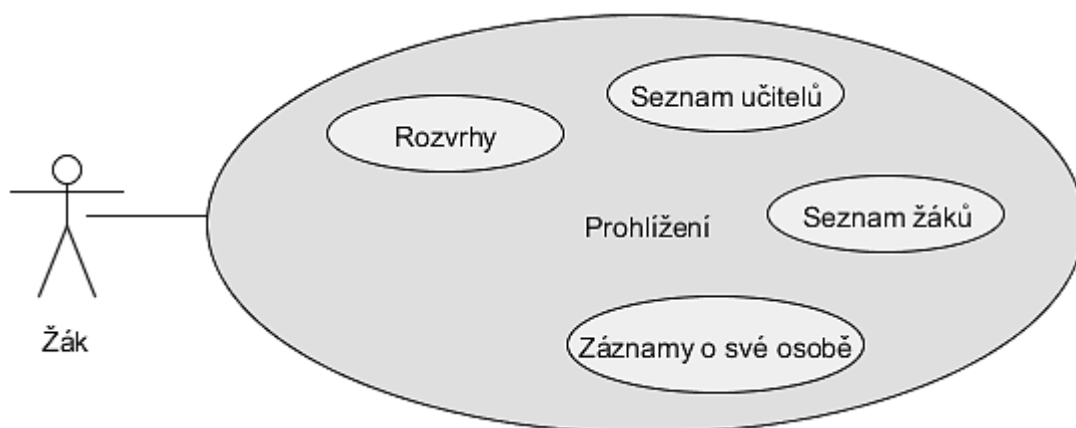
Obrázek 1, use case anonymní

Za anonymního uživatele se považuje kdokoliv, kdo nespadá do žádné z definovaných rolí (např. návštěvník webových stránek školy).

Prohlížení

Anonymní uživatel má právo prohlížet seznamy žáků a učitelů bez osobních dat. Jedná se tedy o identifikační čísla, jména, příjmení, třídy a v případě učitelů pracovní telefony.

Žák



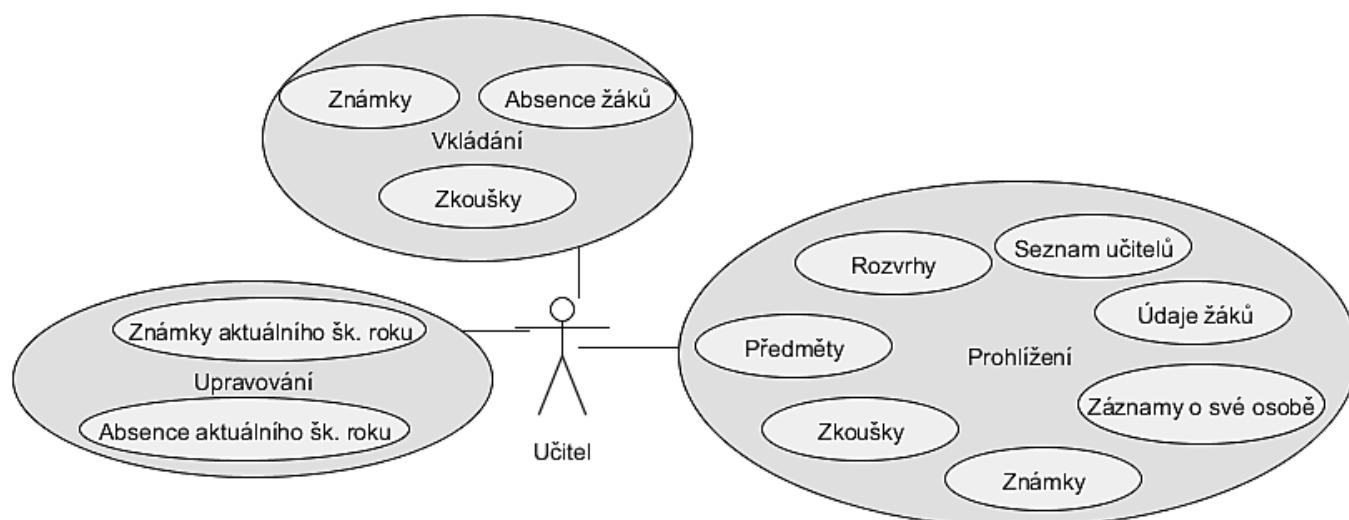
Obrázek 2, use case žák

Uživatelem žák je každý aktivní student školy.

Prohlížení

Žáci si mají stejně jako anonymní uživatelé možnost prohlížet seznamy žáků a učitelů. Dále mohou prohlížet rozvrhy a veškeré informace o sobě a průběhu svého studia.

Učitel



Obrázek 3, use case učitel

Uživatelem učitel je každý pedagogický pracovník školy bez další specializované role.

Prohlížení

Učitelé, stejně jako žáci, mohou prohlížet seznam učitelů a kompletní záznamy o své osobě. Učitelé také mohou prohlížet kompletní údaje o žácích. Dále mohou učitel také prohlížet rozvrhy, předměty a záznamy o zkouškách.

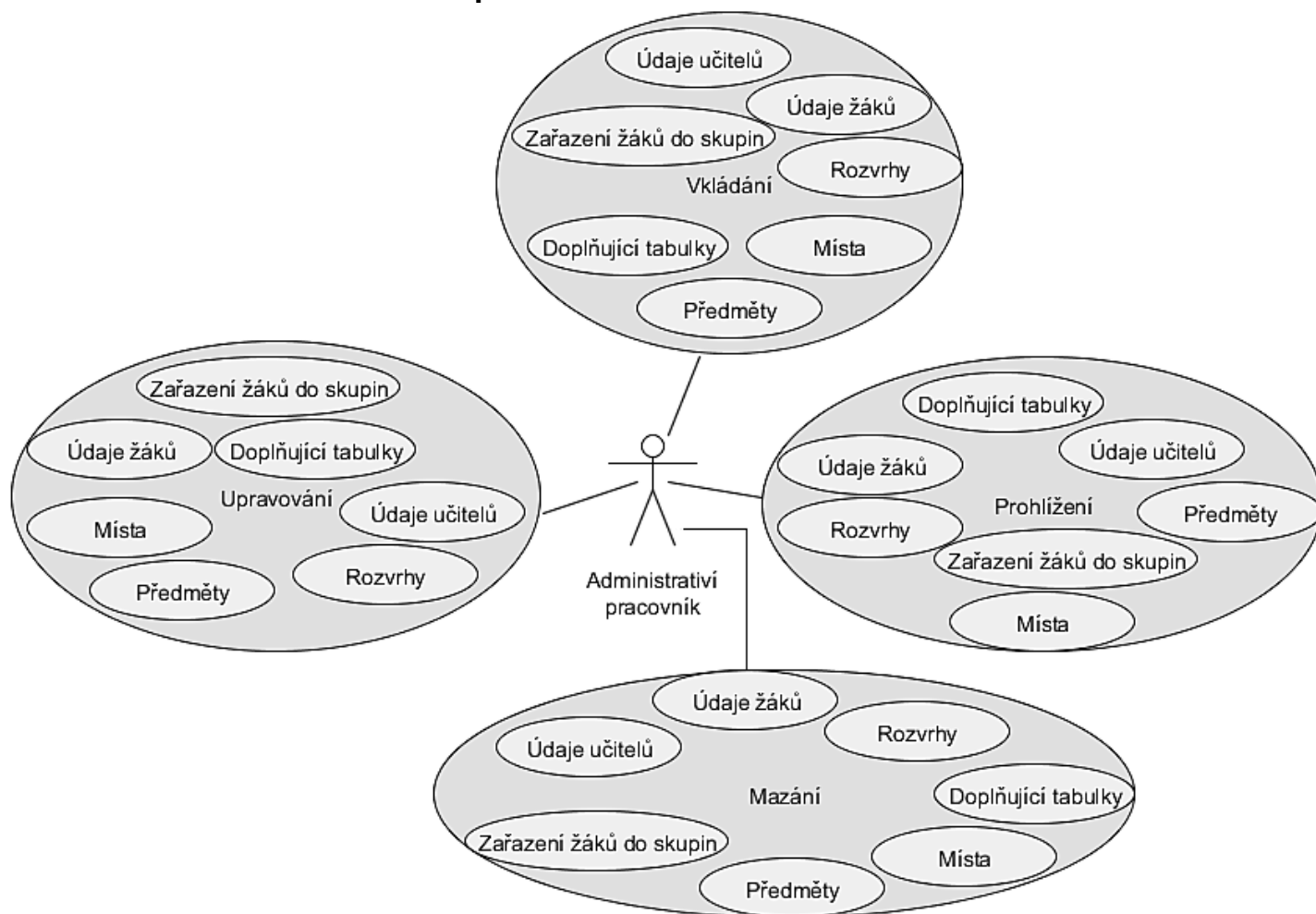
Vkládání

Učitelé mohou vkládat známky a absence žáků, mohou také zadávat záznamy o proběhnutých zkouškách.

Upravování

Učitelé mohou upravovat známky a absence žáků ale pouze aktuálního školního roku.

Administrativní pracovník



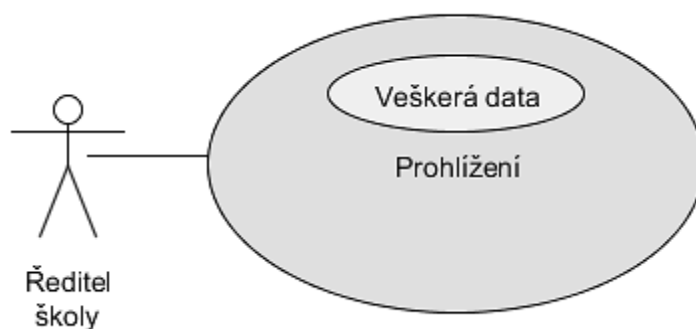
Obrázek 4, use case administrativní pracovník

Uživatelem administrativní pracovník může být učitel nebo se může jednat i o nepedagogického zaměstnance. Administrativní pracovník spravuje data o učitelích a žácích a stará se o tvorbu rozvrhů a tabulky k tomu určené.

Prohlížení, vkládání, upravování a mazání

Administrativní pracovník může prohlížet, vkládat, upravovat a mazat veškeré údaje o žácích a učitelích, předměty, místa pro výuku, rozvrhy, zařazení žáků do skupin a doplňující tabulky, kterými se rozumí typy zkoušek a výchovných opatření a důvody ukončení studia.

Ředitel školy



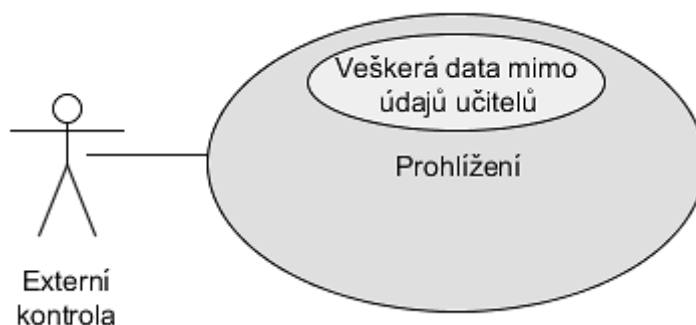
Obrázek 5, use case ředitel školy

Uživatelem ředitel školy je vždy jen jedna osoba. Ředitel školy může být současně i učitelem i administrativním pracovníkem.

Prohlížení

Ředitel školy může prohlížet veškerá data v databázi.

Externí kontrola



Obrázek 6, use case externí kontrola

Uživatel externí kontrola je speciální přístup vytvořený pro státní kontroly školy např. inspekce.

Prohlížení

Externí kontrola může prohlížet veškerá data v databázi mimo osobních údajů učitelů, ke kterým má stejný přístup jako anonymní uživatel.

6.2 Přístupová práva

V první řadě popíši pohledy vytvořené pro usnadnění a upřesnění přístupových práv pro jednotlivé role, následně jednotlivé role a jejich přístupová práva.

Pohledy

Pohled je virtuální tabulkou, která zobrazuje předdefinovaná data z jiných tabulek, ale sama žádná data neukládá. SQL kódy jednotlivých pohledů jsou k nalezení v příloze 10.5.

SEZNAM_ZAKU

Pohled zobrazující ID, jméno, příjmení a třídu všech žáků školy.

SEZNAM_UCITELU

Pohled zobrazující ID, jméno, příjmení, pracovní telefon a třídu, které je třídním učitelem všech učitelů školy.

VLASTNI_UCITEL

Pohled zobrazující učiteli osobní údaje o jeho osobě.

VLASTNI_ZAK

Pohled zobrazující žákovi veškeré informace spjaté s jeho osobou.

UCITEL_ZNAMKY

Pohled zobrazuje známky udělené v aktuálním školním roce.

UCITEL_ABSENCE

Pohled zobrazuje absence zaznamenané v aktuálním školním roce.

Anonymní

Uživatel anonymní není přímo uživatel, nebude mít ani roli, jde o data veřejně dostupná pro zobrazení např. na webových stránkách školy atp.

Pro tyto účely slouží pohledy SEZNAM_ZAKU a SEZNAM_UCITELU které zobrazují neosobní údaje.

Žák

Každý žák bude mít vytvořený vlastní účet. Jednotliví žáci budou zařazeni do role ZAK a jejich uživatelská jména budou tvořena názvem role a jejich ID (např. ZAK1215).

Žáci mají přístup číst seznamy učitelů a žáků, dále mají možnost prohlížet si veškeré údaje o své osobě, k čemuž slouží pohled VLASTNI_ZAK. Mají také možnost prohlížet si rozvrhy.

V rámci databáze to tedy znamená, že role ZAK má právo SELECT na pohledy SEZNAM_ZAKU, SEZNAM_UCITEL a VLASTNI_ZAK a na tabulky PREDMETY, KURZY, HODINY a MISTA.

Učitel

Každý učitel bude mít vytvořený vlastní účet. Jednotliví učitelé budou zařazeni do role UCITEL a jejich uživatelská jména budou tvořena názvem role a jejich ID (např. UCITEL12).

Učitelé mají přístup číst seznamy učitelů, prohlížet si veškeré údaje o své osobě k čemuž slouží pohled VLASTNI_UCITEL, prohlížet údaje o žácích, vyučovaných předmětech, zkouškách, známkách a rozvrzích. Dále mají možnost vkládat známky, zkoušky a absence žáků a upravovat známky a absence aktuálního školního roku.

V rámci databáze to tedy znamená, že role UCITEL má právo SELECT na pohledy SEZNAM_UCITELU, VLASTNÍ_UCITEL, UCITEL_ZNAMKY a UCITEL_ABSENCE a na tabulky ZACI, ZAKONI_ZASTUPCI, ABSENCE, PRERUSENI_STUDIA, OBORY, VYCHOVNA_OPATRENI, ZKOUSKY, PREDMETY, ZNAMKY, KURZY, MISTA a HODINY. Dále právo INSERT do tabulek ZNAMKY, ZKOUSKY a ABSENCE a právo UPDATE na pohledy UCITEL_ZNAMKY a UCITEL_ABSENCE.

Administrativní pracovník

Každý administrativní pracovník bude mít vytvořený vlastní účet. Jednotliví uživatelé budou zařazeni do role ADMINPR.

Administrativní pracovníci mají možnost číst, vkládat a měnit (ve specifických důvodech i mazat) veškerá data o žácích, jejich zákonných zástupcích a učitelích. Starají se také o tvorbu rozvrhů a mají tedy k tabulkám určeným k tomuto účelu plný přístup, spravují také místa pro výuku. Spravují skupiny a zařazují do nich žáky. Starají se také o tabulky spravující typy zkoušek a výchovných opatření a důvody ukončení.

V rámci databáze to tedy znamená, že role ADMINPR bude mít práva SELECT, INSERT, UPDATE a DELETE na tabulky ZACI, UCITELE, ZAKONI_ZASTUPCI, DUVODY_UKONCENI, PRERUSENI_STUDIA, OBORY, VYCHOVNA_OPATRENI, TYPY_VYCH_OPATRENI, MISTA, PREDMETY, TYPY_ZKOUSEK, SKUPINY, KURZY, HODINY, ZARAZENI_ZAKU, PRIRAZENI_PREDMETU a VYUKA_UCITELU.

Ředitel školy

Pro ředitele bude vytvořena role REDITEL hlavně z důvodu zjednodušení předání práv při změně ředitele.

Ředitel školy má možnost prohlížet celou databázi.

V rámci databáze to tedy znamená, že role REDITEL bude mít právo SELECT nad celou databází.

Externí kontrola

Pro kontroly bude vytvořena role KONTROLA.

Pro práva externí kontroly mi nebyla poskytnuta jasná odpověď, tato práva jsou tedy založena na mém pochopení práv inspekce dle školského zákona. [4]

V rámci databáze to tedy znamená, že role KONTROLA bude mít právo SELECT nad celou databází kromě tabulky UCITELE.

7. Závěr

Cílem bakalářské práce bylo vytvořit funkční databázi pro evidenci žáků a učitelů a správu rozvrhů. Databáze je tedy připravena podporovat hlavní funkce aplikace správy žáků a učitelů, záznam a tisk klasifikace a správu rozvrhů. Hlavními přínosy této databáze jsou zefektivnění pro práci učitelů a přístup žákům k databázi. Žáci tak mohou nahlížet na své známky, zkoušky, absence a další údaje, ke kterým v současné době mají přístup jen prostřednictvím učitelů či rodičů přes třídní schůzky.

Na základě informací získaných během mého studia na gymnázium a konzultacemi s učiteli obou zmiňovaných středních škol, které mi výrazně osvětlily některé aspekty funkce střední školy, s kterými jsem se jako žák nesetkal, se mi podařilo daného cíle dosáhnout. Držel jsem se stanoveného postupu, kdy jsem na základě těchto informací vytvořil konceptuální návrh a model, který jsem po následných konzultacích upravil a doplnil tak aby byl co nejjednodušší, ale aby podporoval funkce, které školy využívají ve stávajícím systému. Konceptuální model jsem vytvořil v programu PowerDesigner 12.5, který je sice již staršího data vydání, avšak jsem s tímto programem pracoval během výuky na vysoké škole a měl jsem proto dostatečné zkušenosti v jeho užití k této práci. V PowerDesigneru jsem následně také vygeneroval fyzický model a to konkrétně pro databázový systém Oracle verze 10g jelikož mnou použitá verze 11g nebyla kvůli stáří programu podporována. Fyzický model jsem následně jen minimálně upravil hlavně pro přehlednost a nastavil integritní omezení, která jsem opět konzultoval s jedním z učitelů. I přes zmíněný rozdíl verzí jsem při následném otestování vygenerovaného SQL skriptu v databázovém systému Oracle Database Express Edition 11g Release 2 nenarazil na žádný problém. Testoval jsem základní funkčnost databáze především z pohledu rolí učitele a administrativního pracovníka. Testování proběhlo ručně na malém vzorku smyšlených dat bez použití scénářů vkládáním, výběrem, editací a mazáním záznamů, především k ověření funkčnosti jednotlivých omezení a pohledů, které se ukázaly jako funkční.

Součástí práce bylo také stanovení uživatelů a jejich přístupových práv. I přes doplňující konzultace jsem bohužel nedostal jasné odpovědi na všechny otázky v této oblasti, přesto jsou uživatelé a přístupová práva stanoveny dostatečně pro splnění cílů práce a nemělo by být třeba je měnit.

Databáze byla navrhována pro aplikaci určenou právě pro práci s touto databází. Tato aplikace by při svém použití musela brát ohled na další aplikace, které škola používá, jelikož databáze na základě svého omezení nepokrývá veškerou činnost střední školy. Tvorba této aplikace by byla vhodným budoucím rozšířením této práce stejně jako případné rozšíření databáze pro podporu veškeré činnosti školy.

Databáze byla navržena pro možnost dalšího rozšíření s minimálním zásahem do současné struktury v případě, kdy by vznikl zájem využívat funkce dosud nevyužívané či funkce ani v současném systému nepodporované.

Ačkoli konzultace během tvorby návrhu a po jeho dokončení hodnotily některé části kladně (přístup žáků, spojení do jedné aplikace a odstranění nevyužitých funkcí), jako celek se bohužel oba učitelé shodli, že vzhledem k současnému poměrně sjednocenému stavu systémů pro správu školy na středních školách a s nimi spojených databází je bohužel velice nepravděpodobná implementace této databáze na některé střední škole. Pevně však doufám, že časem školy implementují alespoň nějakou variantu nápadů v této práci a zlepší tak své současné databáze hlavně pro studenty.

8. Terminologický slovník

Termín	Zkratka	Význam [zdroj]
entita		Typ objektů, ať už konkrétních (osoby, místa, věci) či abstraktních (katalogové položky, kategorie), natolik důležitých v dané realitě, že se o takových objektech mají vést záznamy.[9]
atribut		Ekvivalent pole tabulky v relačním modelu.[1] Atributy zaznamenávají vlastnosti entity. [autor]
integritní omezení		Pravidla zajišťující integritu (celistvost) databáze.[autor]
business pravidla		Omezení určitých aspektů databáze založené na způsobu jak organizace chápe a používá data.[1]
structured query language	SQL	Strukturovaný dotazovací jazyk dotazovací jazyk používaný pro práci s daty v relačních databázích.[10]
use case diagram		Diagram zobrazující interakci mezi rolí a systémem.[autor]

9. Zdroje

- [1] HERNANDEZ, Michael J. Návrh Databází. 2. vydání. Praha : Grada Publishing, 2006. 408 s. ISBN 80-247-0900-7
- [2] FOWLER, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition. Boston : Addison-Wesley, 2004. 175 s. ISBN 0-321-19368-7
- [3] SYBASE. SyBooks™ Online [online]. poslední změna duben 2007 [cit. 2011-11-19]. PowerDesigner 12.5 Conceptual Data Model. Dostupné z WWW: http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.stf.powerdesigner.doc_12.5.0/html/cdug/meta.htm
- [4] MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY [online]. [cit. 2011-11-22]. Úplné znění školského zákona. Dostupné z WWW: <http://www.msmt.cz/dokumenty/uplne-zneni-zakona-c-561-2004-sb>
- [5] Wikipedie: Otevřená encyklopedie [online]. poslední změna 6.10.2011 [cit. 2011-11-23]. Primární klíč. Dostupné z WWW: http://cs.wikipedia.org/wiki/Primární_klíč
- [6] Wikipedie: Otevřená encyklopedie [online]. poslední změna 26.3.2011 [cit. 2011-11-23]. Cizí klíč. Dostupné z WWW: http://cs.wikipedia.org/wiki/Cizí_klíč
- [7] PALOVSKÁ, Helena. Krokodýlvy databáze [online]. poslední změna 18.05.2011 [cit. 2011-11-21]. Power Designer – entity a atributy. Dostupné z WWW: <http://krokodata.vse.cz/DM/PDPrimer>
- [8] ORACLE [online]. [cit. 2011-11-23]. Oracle Database Online Documentation 11g Release 2 (11.2). Dostupné z WWW: <http://www.oracle.com/pls/db112/homepage>
- [9] PALOVSKÁ, Helena. Krokodýlvy databáze [online]. poslední změna 24.05.2011 [cit. 2011-12-2]. Entity a vztahy mezi nimi. Dostupné z WWW: <http://krokodata.vse.cz/DM/ERA>
- [10] Wikipedie: Otevřená encyklopedie [online]. poslední změna 22.10.2011 [cit. 2011-12-4]. SQL. Dostupné z WWW: <http://cs.wikipedia.org/wiki/SQL>
- [11] PACHNER, vzdělávací software [online]. poslední změna 6.3.2012 [cit. 2012-3-24]. BAKALÁŘI, systém pro administrativu školy (tisk vysvědčení, evidence, školní matrika, rozvrh ...). Dostupné z WWW: <http://www.pachner.cz/bakalari/bakalari.htm>
- [12] Metodický portál RVP.CZ [online]. publikováno 17.03.2010 [cit. 2012-5-3]. Školní informační systémy. Dostupné z WWW: <http://clanky.rvp.cz/clanek/c/Z/8019/skolni-informacni-systemy.html/>
- [13] Integrovaný studijní informační systém VŠE [online]. [cit. 2012-5-7]. Závěrečné práce. Dostupné z WWW: https://isis.vse.cz/zp/portal_zp.pl?podrobnosti=108050
- [14] Portál UTB [online]. [cit. 2012-5-7]. Kvalifikační práce. Dostupné z WWW: https://portal.utb.cz/wps/portal/!ut/p/c5/04_SB8K8xLLM9MSSzPy8xBz9CP0os3iXQH_vAGcLEwMLSzMjA08DPxNnb19_AwMLY6B8pFm8s7ujh4m5j4GBv1GYgYGRn2lwoEFosLGBpzEB3eEq-DrB8kb4ACOBvp-Hvm5qfoFuREGWSaOigCl-4mS/dl3/d3/L0IDU0IKSkpDZ3BSQ1FvS1VRb0tVUW9LVVFrZyEvWUdVSUFBSUIJSU1NS0VFOUFDR09HT0NH_SUJKRkpGQkpORE5EQk5MSExlQkxBb0VBUFBBS80QzFiOVdfTnlwUkprQWxJTVJKa1FsTUtUSVJLVVtUmlV_NGIKSDQhLzdfRFFPS1BDODQwODk2MjBJME40Q0tNTzBLMjQvaWJtLmludi8xOTc2NDAXODYyMjEvcHJvaGxpemVuaUFjdGlvb9jei56Y3Uuc3RhZy5wb3J0bGV0czE2OC5wcm9obGl6ZW5pLnByYWNILiByYWNIRGV0YWIsQWN0aW9uL3ByYWNISWRuby8yMDc5OS9kZXRhWwvcHJhY2VJbmZv/

- [15] Informační systém Masarykovy univerzity [online]. poslední změna 22.6.2011 [cit. 2012-5-7]. Archiv závěrečné práce Maroš Džoganík FI B-AP BcAP, učo 207841. Dostupné z WWW: http://is.muni.cz/th/207841/fi_b/bc_praca.pdf?info=1
- [16] Katalog vysokoškolských kvalifikačních prací Archivu UHK [online]. [cit. 2012-5-7]. Detail práce eB1216. Dostupné z WWW: <http://evskp.uhk.cz/eB1216>

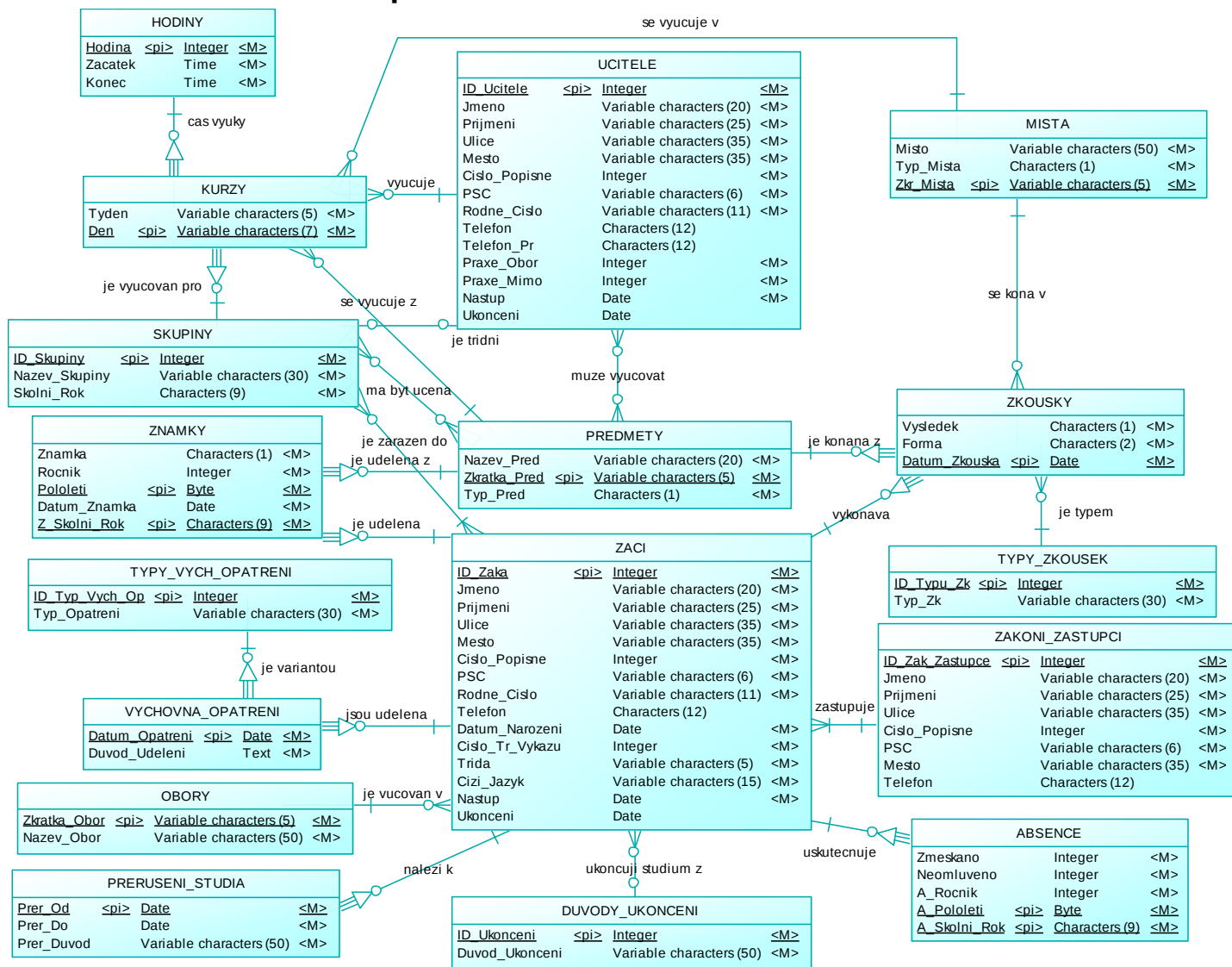
10. Přílohy

10.1 Tabulky

Funkce Entity	evidence žáků, jejich zákonných zástupců, absencí a výchovných opatření	evidence učitelů	zpracování pololetní klasifikace, tvorba a tisk vysvědčení	zpracování mimopololetních zkoušek (maturity, reparáty)	tvorba a tisk rozhřů	sdělování informací o studiu žákům
HODINY					X	Funkce není mapována na entitu, viz příslušná kapitola
UCITELE		X			X	
MISTA				X	X	
KURZY					X	
SKUPINY					X	
PREDMETY			X	X	X	
ZKOUSKY				X		
TYPY_ZKOUSEK				X		
ZNAMKY			X			
ZACI	X		X	X		
ZAKONI_ZASTUPCI	X					
ABSENCE	X					
DUVODY_UKONCENI	X					
PRERUSENI_STUDIA	X					
OBORY	X					
VYCHOVNA_OPATR ENI	X					
TYPY_VYCH_OPATR ENI	X					

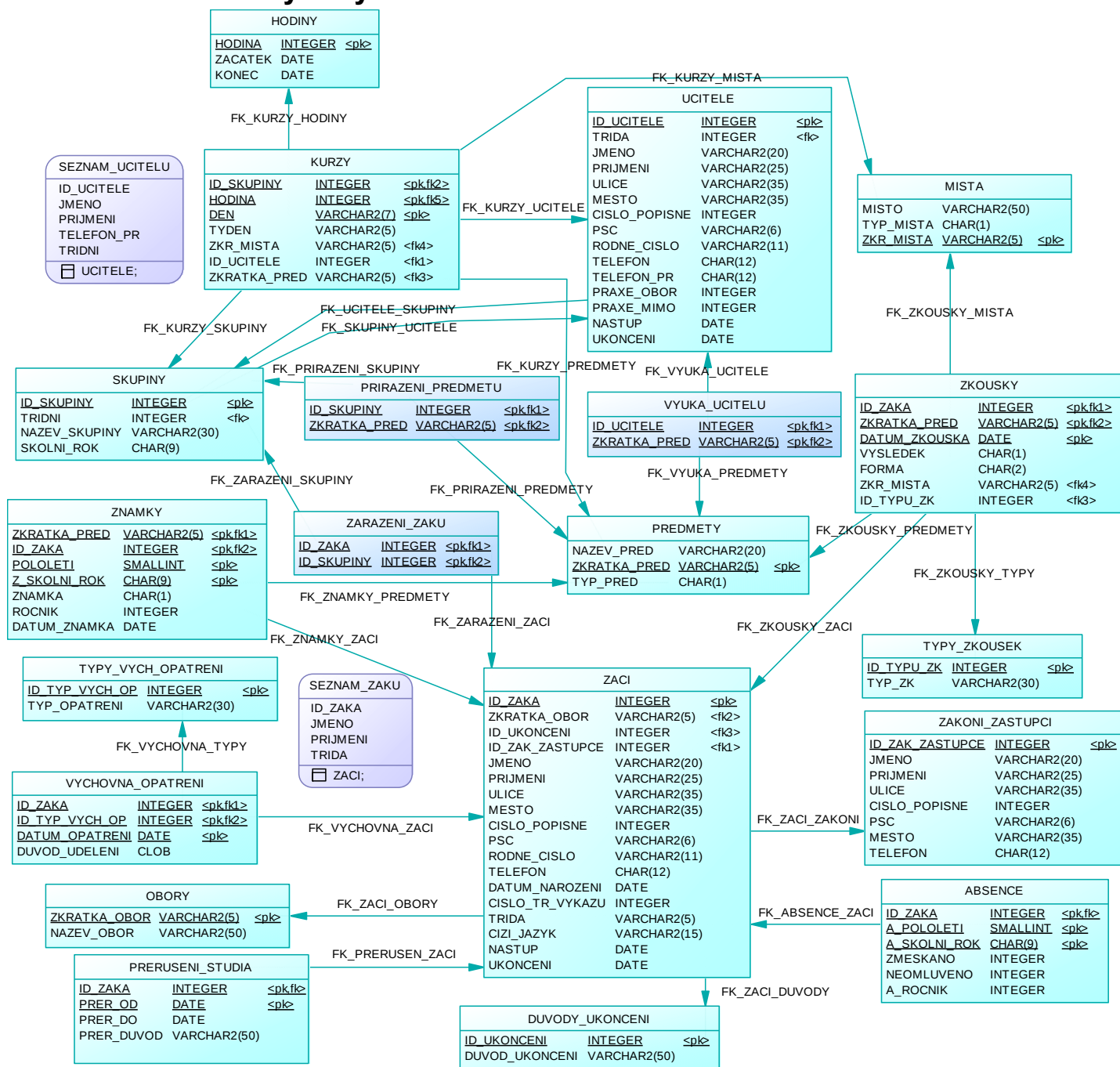
Tabulka 2, mapování entit

10.2 Konceptuální model



Obrázek 7, konceptuální model

10.3 Fyzický model



Obrázek 8, fyzický model

10.4 SQL skript

```
/*=====*/
/* DBMS name:      ORACLE Version 10g      */
/* Created on:      23.3.2012 17:50:55      */
/*=====*/
```

```
alter table ABSENCE
  drop constraint FK_ABSENCE_ZACI;

alter table KURZY
  drop constraint FK_KURZY_HODINY;

alter table KURZY
  drop constraint FK_KURZY_SKUPINY;

alter table KURZY
  drop constraint FK_KURZY_MISTA;

alter table KURZY
  drop constraint FK_KURZY_PREDMETY;

alter table KURZY
  drop constraint FK_KURZY_UCITELE;

alter table PRERUSENI_STUDIA
  drop constraint FK_PRERUSEN_ZACI;

alter table PRIRAZENI_PREDMETU
  drop constraint FK_PRIRAZENI_SKUPINY;

alter table PRIRAZENI_PREDMETU
  drop constraint FK_PRIRAZENI_PREDMETY;

alter table SKUPINY
  drop constraint FK_SKUPINY_UCITELE;

alter table UCITELE
  drop constraint FK_UCITELE_SKUPINY;

alter table VYCHOVNA_OPATRENI
  drop constraint FK_VYCHOVNA_TYPY;

alter table VYCHOVNA_OPATRENI
  drop constraint FK_VYCHOVNA_ZACI;

alter table VYUKA_UCITELU
  drop constraint FK_VYUKA_UCITELE;

alter table VYUKA_UCITELU
  drop constraint FK_VYUKA_PREDMETY;

alter table ZACI
  drop constraint FK_ZACI_OBORY;

alter table ZACI
  drop constraint FK_ZACI_DUVODY;

alter table ZACI
```

```

drop constraint FK_ZACI_ZAKONI;

alter table ZARAZENI_ZAKU
drop constraint FK_ZARAZENI_ZACI;

alter table ZARAZENI_ZAKU
drop constraint FK_ZARAZENI_SKUPINY;

alter table ZKOUSKY
drop constraint FK_ZKOUSKY_PREDMETY;

alter table ZKOUSKY
drop constraint FK_ZKOUSKY_TYPY;

alter table ZKOUSKY
drop constraint FK_ZKOUSKY_MISTA;

alter table ZKOUSKY
drop constraint FK_ZKOUSKY_ZACI;

alter table ZNAMKY
drop constraint FK_ZNAMKY_ZACI;

alter table ZNAMKY
drop constraint FK_ZNAMKY_PREDMETY;

drop table ABSENCE cascade constraints;

drop table DUVODY_UKONCENI cascade constraints;

drop table HODINY cascade constraints;

drop table KURZY cascade constraints;

drop table MISTA cascade constraints;

drop table OBORY cascade constraints;

drop table PREDMETY cascade constraints;

drop table PRERUSENI_STUDIA cascade constraints;

drop table PRIRAZENI_PREDMETU cascade constraints;

drop table SKUPINY cascade constraints;

drop table TYPY_VYCH_OPATRENI cascade constraints;

drop table TYPY_ZKOUSEK cascade constraints;

drop table UCITELE cascade constraints;

drop table VYCHOVNA_OPATRENI cascade constraints;

drop table VYUKA_UCITELU cascade constraints;

drop table ZACI cascade constraints;

drop table ZAKONI_ZASTUPCI cascade constraints;

```

```

drop table ZARAZENI_ZAKU cascade constraints;

drop table ZKOUSKY cascade constraints;

drop table ZNAMKY cascade constraints;

/*=====*/
/* Table: ABSENCE */
/*=====*/
create table ABSENCE (
    "ID_Zaka"            INTEGER                not null,
    "A_Pololeti"         SMALLINT              not null,
    "A_Skolni_Rok"       CHAR(9)               not null,
    "Zmeskano"           INTEGER                not null,
    "Neomluveno"         INTEGER                not null,
    "A_Rocnik"           INTEGER                not null,
    constraint PK_ABSENCE primary key ("ID_Zaka", "A_Pololeti", "A_Skolni_Rok")
);

alter table ABSENCE
    add constraint CKT_ABSENCE check ("Neomluveno" <= "Zmeskano");

alter table ABSENCE
    add constraint CKC_ID_ZAKA_ABSENCE check ("ID_Zaka" >= 1000);

alter table ABSENCE
    add constraint CKC_A_POLOLETI_ABSENCE check ("A_Pololeti" between 1 and 2);

alter table ABSENCE
    add constraint CKC_A_ROCNIK_ABSENCE check ("A_Rocnik" between 1 and 8);

/*=====*/
/* Table: DUVODY_UKONCENI */
/*=====*/
create table DUVODY_UKONCENI (
    "ID_Ukonceni"        INTEGER                not null,
    "Duvod_Ukonceni"     VARCHAR2(50)          not null,
    constraint PK_DUVODY_UKONCENI primary key ("ID_Ukonceni")
);

/*=====*/
/* Table: HODINY */
/*=====*/
create table HODINY (
    "Hodina"             INTEGER                not null,
    "Zacatek"            DATE                  not null,
    "Konec"              DATE                  not null,
    constraint PK_HODINY primary key ("Hodina")
);

alter table HODINY
    add constraint CKT_HODINY check ("Konec" > "Zacatek");

/*=====*/
/* Table: KURZY */
/*=====*/
create table KURZY (
    "ID_Skupiny"         INTEGER                not null,

```

```

        "Hodina"                INTEGER                not null,
        "Den"                   VARCHAR2(7)            not null,
        "Tyden"                 VARCHAR2(5)            not null,
        "Zkr_Mista"             VARCHAR2(5)            not null,
        "ID_Ucitele"            INTEGER                not null,
        "Zkratka_Pred"          VARCHAR2(5)            not null,
        constraint PK_KURZY primary key ("ID_Skupiny", "Hodina", "Den")
    );

alter table KURZY
    add constraint CKC_DEN_KURZY check ("Den" in
('PONDELI','UTERY','STREDA','CTVRTEK','PATEK'));

alter table KURZY
    add constraint CKC_TYDEN_KURZY check ("Tyden" in ('LICHY','SUDY','OBA'));

alter table KURZY
    add constraint CKC_ID_UCITELE_KURZY check ("ID_Ucitele" <= 999);

alter table KURZY
    add constraint CKC_ZKRATKA_PRED_KURZY check ("Zkratka_Pred" =
upper("Zkratka_Pred"));

/*=====*/
/* Table: MISTA */
/*=====*/
create table MISTA (
    "Misto"                VARCHAR2(50)                not null,
    "Typ_Mista"            CHAR(1)                    not null,
    "Zkr_Mista"            VARCHAR2(5)                not null,
    constraint PK_MISTA primary key ("Zkr_Mista")
);

alter table MISTA
    add constraint CKC_TYP_MISTA_MISTA check ("Typ_Mista" in ('S','M'));

/*=====*/
/* Table: OBORY */
/*=====*/
create table OBORY (
    "Zkratka_Obor"        VARCHAR2(5)                not null,
    "Nazev_Obor"          VARCHAR2(50)                not null,
    constraint PK_OBORY primary key ("Zkratka_Obor")
);

alter table OBORY
    add constraint CKC_ZKRATKA_OBOR_OBORY check ("Zkratka_Obor" =
upper("Zkratka_Obor"));

/*=====*/
/* Table: PREDMETY */
/*=====*/
create table PREDMETY (
    "Nazev_Pred"          VARCHAR2(20)                not null,
    "Zkratka_Pred"        VARCHAR2(5)                not null,
    "Typ_Pred"            CHAR(1)                    not null,
    constraint PK_PREDMETY primary key ("Zkratka_Pred")
);

```

```

alter table PREDMETY
    add constraint CKC_ZKRATKA_PRED_PREDMETY check ("Zkratka_Pred" =
upper("Zkratka_Pred"));

alter table PREDMETY
    add constraint CKC_TYP_PRED_PREDMETY check ("Typ_Pred" in ('P','V','J'));

/*=====*/
/* Table: PRERUSENI_STUDIA */
/*=====*/
create table PRERUSENI_STUDIA (
    "ID_Zaka"            INTEGER                not null,
    "Prer_Od"            DATE                    not null,
    "Prer_Do"            DATE                    not null,
    "Prer_Duvod"          VARCHAR2(50)          not null,
    constraint PK_PRERUSENI_STUDIA primary key ("ID_Zaka", "Prer_Od")
);

alter table PRERUSENI_STUDIA
    add constraint CKT_PRERUSENI_STUDIA check ("Prer_Do" > "Prer_Od");

alter table PRERUSENI_STUDIA
    add constraint CKC_ID_ZAKA_PRERUSEN check ("ID_Zaka" >= 1000);

/*=====*/
/* Table: PRIRAZENI_PREDMETU */
/*=====*/
create table PRIRAZENI_PREDMETU (
    "ID_Skupiny"          INTEGER                not null,
    "Zkratka_Pred"        VARCHAR2(5)           not null,
    constraint PK_PRIRAZENI_PREDMETU primary key ("ID_Skupiny", "Zkratka_Pred")
);

alter table PRIRAZENI_PREDMETU
    add constraint CKC_ZKRATKA_PRED_PRIRAZEN check ("Zkratka_Pred" =
upper("Zkratka_Pred"));

/*=====*/
/* Table: SKUPINY */
/*=====*/
create table SKUPINY (
    "ID_Skupiny"          INTEGER                not null,
    "Tridni"              INTEGER,
    "Nazev_Skupiny"        VARCHAR2(30)          not null,
    "Skolni_Rok"          CHAR(9)               not null,
    constraint PK_SKUPINY primary key ("ID_Skupiny")
);

alter table SKUPINY
    add constraint CKC_TRIDNI_SKUPINY check ("Tridni" is null or ("Tridni" <=
999));

/*=====*/
/* Table: TYPY_VYCH_OPATRENI */
/*=====*/
create table TYPY_VYCH_OPATRENI (
    "ID_Typ_Vych_Op"      INTEGER                not null,
    "Typ_Opatreni"         VARCHAR2(30)          not null,
    constraint PK_TYPY_VYCH_OPATRENI primary key ("ID_Typ_Vych_Op")
);

```

```

);

/*=====*/
/* Table: TYPY_ZKOUSEK */
/*=====*/
create table TYPY_ZKOUSEK (
    "ID_Typu_Zk"          INTEGER                not null,
    "Typ_Zk"              VARCHAR2(30)           not null,
    constraint PK_TYPY_ZKOUSEK primary key ("ID_Typu_Zk")
);

/*=====*/
/* Table: UCITELE */
/*=====*/
create table UCITELE (
    "ID_Ucitele"          INTEGER                not null,
    "Trida"               INTEGER,
    "Jmeno"               VARCHAR2(20)           not null,
    "Prijmeni"            VARCHAR2(25)           not null,
    "Ulice"               VARCHAR2(35)           not null,
    "Mesto"               VARCHAR2(35)           not null,
    "Cislo_Popisne"       INTEGER                not null,
    PSC                   VARCHAR2(6)            not null,
    "Rodne_Cislo"         VARCHAR2(11)          not null,
    "Telefon"             CHAR(12),
    "Telefon_Pr"          CHAR(12),
    "Praxe_Obor"          INTEGER                not null,
    "Praxe_Mimo"          INTEGER                not null,
    "Nastup"              DATE                   not null,
    "Ukonceni"            DATE,
    constraint PK_UCITELE primary key ("ID_Ucitele")
);

alter table UCITELE
    add constraint CKT_UCITELE check ("Ukonceni" > "Nastup");

alter table UCITELE
    add constraint CKC_ID_UCITELE_UCITELE check ("ID_Ucitele" <= 999);

alter table UCITELE
    add constraint CKC_PSC_UCITELE check (PSC = upper(PSC));

/*=====*/
/* Table: VYCHOVNA_OPATRENI */
/*=====*/
create table VYCHOVNA_OPATRENI (
    "ID_Zaka"             INTEGER                not null,
    "ID_Typ_Vych_Op"      INTEGER                not null,
    "Datum_Opatreni"       DATE                   not null,
    "Duvod_Udeleni"       CLOB                   not null,
    constraint PK_VYCHOVNA_OPATRENI primary key ("ID_Zaka", "ID_Typ_Vych_Op",
"Datum_Opatreni")
);

alter table VYCHOVNA_OPATRENI
    add constraint CKC_ID_ZAKA_VYCHOVNA check ("ID_Zaka" >= 1000);

/*=====*/
/* Table: VYUKA_UCITELU */
/*=====*/

```

```

/*=====*/
create table VYUKA_UCITELU (
    "ID_Ucitele"          INTEGER                not null,
    "Zkratka_Pred"        VARCHAR2(5)           not null,
    constraint PK_VYUKA_UCITELU primary key ("ID_Ucitele", "Zkratka_Pred")
);

alter table VYUKA_UCITELU
    add constraint CKC_ID_UCITELE_VYUKA_UC check ("ID_Ucitele" <= 999);

alter table VYUKA_UCITELU
    add constraint CKC_ZKRATKA_PRED_VYUKA_UC check ("Zkratka_Pred" =
upper("Zkratka_Pred"));

/*=====*/
/* Table: ZACI */
/*=====*/
create table ZACI (
    "ID_Zaka"              INTEGER                not null,
    "Zkratka_Obor"         VARCHAR2(5)           not null,
    "ID_Ukonceni"          INTEGER,
    "ID_Zak_Zastupce"      INTEGER                not null,
    "Jmeno"                VARCHAR2(20)           not null,
    "Prijmeni"             VARCHAR2(25)           not null,
    "Ulice"                VARCHAR2(35)           not null,
    "Mesto"                VARCHAR2(35)           not null,
    "Cislo_Popisne"        INTEGER                not null,
    PSC                    VARCHAR2(6)            not null,
    "Rodne_Cislo"          VARCHAR2(11)           not null,
    "Telefon"              CHAR(12),
    "Datum_Narozeni"       DATE                  not null,
    "Cislo_Tr_Vykazu"      INTEGER                not null,
    "Trida"                VARCHAR2(5)            not null,
    "Cizi_Jazyk"           VARCHAR2(15)           not null,
    "Nastup"               DATE                  not null,
    "Ukonceni"             DATE,
    constraint PK_ZACI primary key ("ID_Zaka")
);

alter table ZACI
    add constraint CKT_ZACI check ("Ukonceni" > "Nastup");

alter table ZACI
    add constraint CKC_ID_ZAKA_ZACI check ("ID_Zaka" >= 1000);

alter table ZACI
    add constraint CKC_ZKRATKA_OBOR_ZACI check ("Zkratka_Obor" =
upper("Zkratka_Obor"));

alter table ZACI
    add constraint CKC_PSC_ZACI check (PSC = upper(PSC));

alter table ZACI
    add constraint CKC_CISLO_TR_VYKAZU_ZACI check ("Cislo_Tr_Vykazu" >= 1);

/*=====*/
/* Table: ZAKONI_ZASTUPCI */
/*=====*/
create table ZAKONI_ZASTUPCI (

```

```

        "ID_Zak_Zastupce"    INTEGER                not null,
        "Jmeno"              VARCHAR2(20)           not null,
        "Prijmeni"           VARCHAR2(25)           not null,
        "Ulice"              VARCHAR2(35)           not null,
        "Cislo_Popisne"      INTEGER                not null,
        "PSC"                VARCHAR2(6)            not null,
        "Mesto"              VARCHAR2(35)           not null,
        "Telefon"            CHAR(12),
        constraint PK_ZAKONI_ZASTUPCI primary key ("ID_Zak_Zastupce")
    );

alter table ZAKONI_ZASTUPCI
    add constraint CKC_PSC_ZAKONI_Z check (PSC = upper(PSC));

/*=====*/
/* Table: ZARAZENI_ZAKU */
/*=====*/
create table ZARAZENI_ZAKU (
    "ID_Zaka"                INTEGER                not null,
    "ID_Skupiny"             INTEGER                not null,
    constraint PK_ZARAZENI_ZAKU primary key ("ID_Zaka", "ID_Skupiny")
);

alter table ZARAZENI_ZAKU
    add constraint CKC_ID_ZAKA_ZARAZENI check ("ID_Zaka" >= 1000);

/*=====*/
/* Table: ZKOUSKY */
/*=====*/
create table ZKOUSKY (
    "ID_Zaka"                INTEGER                not null,
    "Zkratka_Pred"           VARCHAR2(5)           not null,
    "Datum_Zkouska"          DATE                  not null,
    "Vysledek"               CHAR(1)               not null,
    "Forma"                  CHAR(2)               not null,
    "Zkr_Mista"              VARCHAR2(5)           not null,
    "ID_Typu_Zk"             INTEGER                not null,
    constraint PK_ZKOUSKY primary key ("ID_Zaka", "Zkratka_Pred", "Datum_Zkouska")
);

alter table ZKOUSKY
    add constraint CKC_ID_ZAKA_ZKOUSKY check ("ID_Zaka" >= 1000);

alter table ZKOUSKY
    add constraint CKC_ZKRATKA_PRED_ZKOUSKY check ("Zkratka_Pred" =
upper("Zkratka_Pred"));

alter table ZKOUSKY
    add constraint CKC_VYSLEDEK_ZKOUSKY check ("Vysledek" in
('1','2','3','4','5','N'));

alter table ZKOUSKY
    add constraint CKC_FORMA_ZKOUSKY check ("Forma" in ('PI','US','PR'));

/*=====*/
/* Table: ZNAMKY */
/*=====*/
create table ZNAMKY (
    "Zkratka_Pred"           VARCHAR2(5)           not null,

```

```

        "ID_Zaka"            INTEGER            not null,
        "Pololeti"          SMALLINT           not null,
        "Z_Skolni_Rok"      CHAR(9)            not null,
        "Znamka"            CHAR(1)            not null,
        "Rocnik"            INTEGER            not null,
        "Datum_Znamka"      DATE               not null,
        constraint PK_ZNAMKY primary key ("Zkratka_Pred", "ID_Zaka", "Pololeti",
        "Z_Skolni_Rok")
    );

alter table ZNAMKY
    add constraint CKC_ZKRATKA_PRED_ZNAMKY check ("Zkratka_Pred" =
upper("Zkratka_Pred"));

alter table ZNAMKY
    add constraint CKC_ID_ZAKA_ZNAMKY check ("ID_Zaka" >= 1000);

alter table ZNAMKY
    add constraint CKC_POLOLETI_ZNAMKY check ("Pololeti" between 1 and 2);

alter table ZNAMKY
    add constraint CKC_ZNAMKA_ZNAMKY check ("Znamka" in
('1','2','3','4','5','N','U'));

alter table ZNAMKY
    add constraint CKC_ROCNIK_ZNAMKY check ("Rocnik" between 1 and 8);

alter table ABSENCE
    add constraint FK_ABSENCE_ZACI foreign key ("ID_Zaka")
        references ZACI ("ID_Zaka")
        on delete cascade;

alter table KURZY
    add constraint FK_KURZY_HODINY foreign key ("Hodina")
        references HODINY ("Hodina");

alter table KURZY
    add constraint FK_KURZY_SKUPINY foreign key ("ID_Skupiny")
        references SKUPINY ("ID_Skupiny");

alter table KURZY
    add constraint FK_KURZY_MISTA foreign key ("Zkr_Mista")
        references MISTA ("Zkr_Mista");

alter table KURZY
    add constraint FK_KURZY_PREDMETY foreign key ("Zkratka_Pred")
        references PREDMETY ("Zkratka_Pred");

alter table KURZY
    add constraint FK_KURZY_UCITELE foreign key ("ID_Ucitele")
        references UCITELE ("ID_Ucitele");

alter table PRERUSENI_STUDIA
    add constraint FK_PRERUSEN_ZACI foreign key ("ID_Zaka")
        references ZACI ("ID_Zaka")
        on delete cascade;

alter table PRIRAZENI_PREDMETU
    add constraint FK_PRIRAZENI_SKUPINY foreign key ("ID_Skupiny")

```

```

        references SKUPINY ("ID_Skupiny")
        on delete cascade;

alter table PRIRAZENI_PREDMETU
    add constraint FK_PRIRAZENI_PREDMETY foreign key ("Zkratka_Pred")
        references PREDMETY ("Zkratka_Pred");

alter table SKUPINY
    add constraint FK_SKUPINY_UCITELE foreign key ("Tridni")
        references UCITELE ("ID_Ucitele");

alter table UCITELE
    add constraint FK_UCITELE_SKUPINY foreign key ("Trida")
        references SKUPINY ("ID_Skupiny")
        on delete set null;

alter table VYCHOVNA_OPATRENI
    add constraint FK_VYCHOVNA_TYPY foreign key ("ID_Typ_Vych_Op")
        references TYPY_VYCH_OPATRENI ("ID_Typ_Vych_Op");

alter table VYCHOVNA_OPATRENI
    add constraint FK_VYCHOVNA_ZACI foreign key ("ID_Zaka")
        references ZACI ("ID_Zaka")
        on delete cascade;

alter table VYUKA_UCITELU
    add constraint FK_VYUKA_UCITELE foreign key ("ID_Ucitele")
        references UCITELE ("ID_Ucitele")
        on delete cascade;

alter table VYUKA_UCITELU
    add constraint FK_VYUKA_PREDMETY foreign key ("Zkratka_Pred")
        references PREDMETY ("Zkratka_Pred")
        on delete cascade;

alter table ZACI
    add constraint FK_ZACI_OBORY foreign key ("Zkratka_Obor")
        references OBORY ("Zkratka_Obor");

alter table ZACI
    add constraint FK_ZACI_DUVODY foreign key ("ID_Ukonceni")
        references DUVODY_UKONCENI ("ID_Ukonceni");

alter table ZACI
    add constraint FK_ZACI_ZAKONI foreign key ("ID_Zak_Zastupce")
        references ZAKONI_ZASTUPCI ("ID_Zak_Zastupce");

alter table ZARAZENI_ZAKU
    add constraint FK_ZARAZENI_ZACI foreign key ("ID_Zaka")
        references ZACI ("ID_Zaka")
        on delete cascade;

alter table ZARAZENI_ZAKU
    add constraint FK_ZARAZENI_SKUPINY foreign key ("ID_Skupiny")
        references SKUPINY ("ID_Skupiny")
        on delete cascade;

alter table ZKOUSKY
    add constraint FK_ZKOUSKY_PREDMETY foreign key ("Zkratka_Pred")

```

```

        references PREDMETY ("Zkratka_Pred");

alter table ZKOUSKY
    add constraint FK_ZKOUSKY_TYPY foreign key ("ID_Typu_Zk")
        references TYPY_ZKOUSEK ("ID_Typu_Zk");

alter table ZKOUSKY
    add constraint FK_ZKOUSKY_MISTA foreign key ("Zkr_Mista")
        references MISTA ("Zkr_Mista");

alter table ZKOUSKY
    add constraint FK_ZKOUSKY_ZACI foreign key ("ID_Zaka")
        references ZACI ("ID_Zaka")
        on delete cascade;

alter table ZNAMKY
    add constraint FK_ZNAMKY_ZACI foreign key ("ID_Zaka")
        references ZACI ("ID_Zaka")
        on delete cascade;

alter table ZNAMKY
    add constraint FK_ZNAMKY_PREDMETY foreign key ("Zkratka_Pred")
        references PREDMETY ("Zkratka_Pred");

```

10.5 Pohledy

Generované

```

/*=====*/
/* View: SEZNAM_UCITELU */
/*=====*/
create or replace view SEZNAM_UCITELU as
select "ID_Ucitele", "Jmeno", "Prijmeni", "Telefon_Pr", "Tridni"
from UCITELE;

/*=====*/
/* View: SEZNAM_ZAKU */
/*=====*/
create or replace view SEZNAM_ZAKU as
select "ID_Zaka", "Jmeno", "Prijmeni", "Trida" from ZACI;

```

Vlastní

VLASTNI_UCITEL

```

create view VLASTNi_UCITEL as
select * from ucitele where ('UCITEL' || "ID_Ucitele") like (select user
from dual);

```

VLASTNI_ZAK

```

create view VLASTNi_ZAK as
select * from zaci join zakoni_zastupci using("ID_Zak_Zastupce")
join absence using("ID_Zaka") join duvody_ukonceni using("ID_Ukonceni")
join preruseni_studia using("ID_Zaka") join obory using("Zkratka_Obor")
join vychovna_opatreni using("ID_Zaka")
join typy_vych_opatreni using("ID_Typ_Vych_Op")
join znamky using("ID_Zaka") join zkousky using("ID_Zaka")
join typy_zkousek using ("ID_Typu_Zk")

```

```
where ('ZAK' || "ID_Zaka") like (select user from dual);
```

UCITEL_ZNAMKY

```
create view UCITEL_ZNAMKY as
select * from znamky where "Z_Skolni_Rok" like
(to_char(add_months(sysdate,-8), 'YYYY') || '/' ||
to_char(add_months(sysdate,4), 'YYYY'));
```

UCITEL_ABSENCE

```
create view UCITEL_ABSENCE as
select * from absence where "A_Skolni_Rok" like
(to_char(add_months(sysdate,-8), 'YYYY') || '/' ||
to_char(add_months(sysdate,4), 'YYYY'));
```

10.6 CHECK a Trigger

CHECK vložení důvodu ukončení

```
alter table ZACI add constraint CKC_DUVODY
CHECK ("Ukonceni" is not null and "ID_Ukonceni" is not null or "Ukonceni"
is null);
```

Ověření začátku přerušení

```
create or replace
TRIGGER OVERENI_Z_PRERUSENI
BEFORE INSERT OR UPDATE ON PRERUSENI_STUDIA
FOR EACH ROW
declare
nastup date;
ukonceni date;
BEGIN
  begin
    select zaci."Nastup" into nastup from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  begin
    select zaci."Ukonceni" into ukonceni from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  IF :new."Prer_Od"-nastup < 0 or :new."Prer_Od"-ukonceni > 0 then
    RAISE_APPLICATION_ERROR(-20001, 'Zacatek preruseni studia musi byt behem
studia zaka');
  end if;
END;
```

Ověření ukončení přerušení

```
create or replace
TRIGGER OVERENI_K_PRERUSENI
BEFORE INSERT OR UPDATE ON PRERUSENI_STUDIA
FOR EACH ROW
declare
nastup date;
ukonceni date;
```

```

BEGIN
  begin
    select zaci."Nastup" into nastup from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  begin
    select zaci."Ukonceni" into ukonceni from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  IF :new."Prer_Do"-nastup < 0 or :new."Prer_Do"-ukonceni > 0 then
    RAISE_APPLICATION_ERROR(-20002, 'Konec preruseni studia musi byt behem
studia zaka');
  end if;
END;

```

Ověření data výchovného opatření

```

create or replace
TRIGGER OVERENI_DATA_VO
BEFORE INSERT OR UPDATE ON VYCHOVNA_OPATRENI
FOR EACH ROW
declare
nastup date;
ukonceni date;
BEGIN
  begin
    select zaci."Nastup" into nastup from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  begin
    select zaci."Ukonceni" into ukonceni from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  IF :new."Datum_Opatreni"-nastup < 0 or :new."Datum_Opatreni"-ukonceni > 0
then
    RAISE_APPLICATION_ERROR(-20003, 'Vychovne opatreni musi byt udeleno behem
studia zaka');
  end if;
END;

```

Ověření data známky

```

create or replace
TRIGGER OVERENI_DATA_ZN
BEFORE INSERT OR UPDATE ON ZNAMKY
FOR EACH ROW
declare
nastup date;
ukonceni date;
BEGIN
  begin
    select zaci."Nastup" into nastup from zaci where
      :new."ID_Zaka" = zaci."ID_Zaka";
  end;
  begin
    select zaci."Ukonceni" into ukonceni from zaci where

```

```

        :new."ID_Zaka" = zaci."ID_Zaka";
    end;
    IF :new."Datum_Znamka"-nastup < 0 or :new."Datum_Znamka"-ukonceni > 0
then
    RAISE_APPLICATION_ERROR(-20004, 'Znamka musi byt udelena behem studia
zaka');
    end if;
END;

```

Ověření data zkoušky

```

create or replace
TRIGGER OVERENI_DATA_ZK
BEFORE INSERT OR UPDATE ON ZKOUSKY
FOR EACH ROW
declare
nastup date;
ukonceni date;
BEGIN
    begin
        select zaci."Nastup" into nastup from zaci where
        :new."ID_Zaka" = zaci."ID_Zaka";
    end;
    begin
        select zaci."Ukonceni" into ukonceni from zaci where
        :new."ID_Zaka" = zaci."ID_Zaka";
    end;
    IF :new."Datum_Zkouska"-nastup < 0 or :new."Datum_Zkouska"-ukonceni > 0
then
    RAISE_APPLICATION_ERROR(-20005, 'Datum konani zkousky musi byt behem
studia zaka');
    end if;
END;

```