VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH SQL DATABÁZE PRO STŘEDNÍ PRŮMYSLOVOU ŠKOLU FRÝDEK-MÍSTEK

PROPOSAL OF SQL DATABASE FOR HIGH TECHNICAL SCHOOL IN FRÝDEK-MÍSTEK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ BRET
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KŘÍŽ, Ph.D.
SUPERVISOR

BRNO 2014

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2013/2014
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Bret Ondřej

Manažerská informatika (6209R021)

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

Návrh SQL databáze pro Střední průmyslovou školu Frýdek-Místek

v anglickém jazyce:

Proposal of SQL Database for High Technical School in Frýdek-Místek

Pokyny pro vypracování:

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

Seznam odborné literatury:

KOCH, M. Datové a funkční modelování. Brno: CERM, 2004.
ISBN 80-214-2724-8.
LACKO, L. 1001 tipů a triků pro SQL. Brno: Computer Press, 2011.
ISBN 978-80-251-3010-0
OPPEL, A. SQL bez předchozích znalostí. Brno: Computer Press, 2012.
ISBN 978-80-251-1707-1
STEPHENS, R.,R. PLEW a A. D. JONES. Naučte se SQL za 28 dní. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1

Vedoucí bakalářské práce: Ing. Jiří Kříž, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2013/2014.

L.S.

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

V Brně, dne 28.05.2014

Abstrakt

Obsahem této bakalářské práce je návrh SQL databáze pro Střední průmyslovou školu ve Frýdku-Místku. Práce je rozdělena na teoretickou část a praktickou část. V teoretické části jsou uvedeny potřebné informace a teorie nezbytná k úspěšnému splnění cíle práce. V praktické části je analyzována současná situace školy a je navrženo vlastní řešení daného problému.

Abstract

The content of this bachelor thesis is to design SQL database for the Secondary Technical School in Frýdek-Místek. The work is divided into a theoretical part and a practical part. The theoretical part provides the necessary information and theories important for successful fulfilling the goal of the work. The practical part analyses the current situation of the school and offers a solution to the problem mentioned.

Klíčová slova

SQL, primární klíč, databáze, škola, datové modely, normalizace, dotaz, procedura

Key words

SQL, primary key, database, school, data models, normalization, query, procedure

Bibliografická citace práce

BRET, O. *Návrh SQL databáze pro střední průmyslovou školu Frýdek-Místek.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 58s. Vedoucí bakalářské práce Ing. Jiří Kříž, Ph.D.

Čestné prohlášení

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

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

V Brně dne 28. 5. 2014 ……

Podpis studenta

Poděkování

Rád bych tímto poděkoval vedoucímu bakalářské práce panu Ing. Jiřímu Křížovi, Ph.D. za vstřícný přístup, cenné rady a připomínky, které mi pomohly při řešení této práce.

Dále bych rád poděkoval paní Ing. Jiřině Beňové za poskytnutí podkladů pro mou práci a její vstřícný přístup.

Obsah

Úvod

Každým dnem přibývá na Zemi ohromné množství dat. Začíná být stále obtížnější tyto data zpracovat a poté z těchto dat získat relevantní informace. Ukládání těchto dat v papírové podobě se stalo neefektivním. Proto vznikly elektronické databáze. V dnešním světě jsou elektronické databáze běžnou součástí našeho každodenního života. Usnadňují a urychlují nám práci s ohromným množstvím dat. Právě s tímto zastaralým a neefektivním ukládáním dat se můžeme setkat v českém školství, které trpí dlouhodobým podfinancováním. Z důvodu velkého množství uchovávaných dat by mělo jít s dobou a využívat nejmodernější dostupné technologie.

Přechod na elektronické databáze usnadní vyhledávání informací, zrychlí a celkově zefektivní práci s těmito daty. Tento přechod k moderním technologiím je zpomalován nezkušeností, neznalostí ze strany učitelů a finančními náklady, které si plno škol nemůže z rozpočtových důvodů dovolit. Škola, se kterou spolupracuji, patří k největším v okrese Frýdek-Místek s celkovým počtem žáků přes 600.

1. Vymezení problémů a cíl práce

Cílem mé práce je návrh databáze, která bude archivovat hodnocení studentů za dobu jejich studia na Střední průmyslové škole ve Frýdku-Místku. Důraz je kladen na uchovávání údajů, které se používají pro výpis maturitního protokolu a uchovávání údajů o žákovských pracích, které se archivují 5 let a v případě maturitního protokolu let 45.

2. Teoretická východiska práce

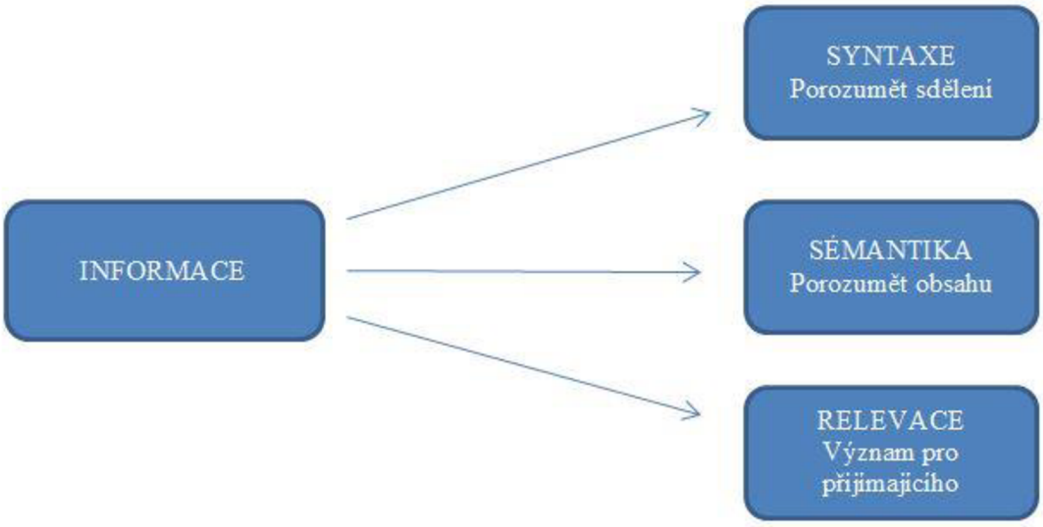
V této části popíši nezbytnou základní teorii, bez které by nebylo možné dosáhnout zvoleného cíle. Jako první charakterizuji základní pojmy a pojem databáze obecně. Poté popíši jednotlivé kroky a postupy tvory nové databáze a nakonec stručně popíši jazyk SQL, ve kterém budu databázi tvořit a jeho základní příkazy.

2.1 Co jsou to informace, data a znalosti

V této kapitole si vyjasníme pojmy informace, data a znalosti a jak s nimi nakládat.

2.1.1 Informace

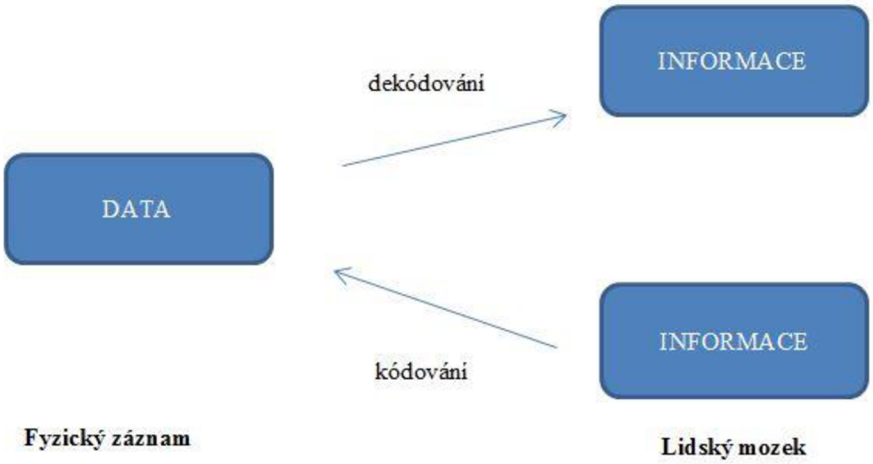
Na informaci se můžeme dívat z různých úhlů. Můžeme na ní nahlížet jako na zprávu nebo vjem, který splňuje 3 podmínky. První podmínkou je syntaktická relevance. Přijímající subjekt musí zprávu detekovat a musí jí rozumět. Druhou podmínkou je sémantická relevance. Přijímající subjekt musí vědět, co zpráva znamená a co svědčí o něm a jeho okolí. Třetí podmínkou je pragmatická relevance. Pro přijímající subjekt musí mít zpráva nějaký relevantní význam. Informace usnadňuje naše rozhodování [1].



Obrázek č. 1: Informace (Zdroj: převzato z 1, s. 4)

2.1.2 Data

Data je výraz pro údaje popisující nějaký jev nebo objekt. Když se člověk rozhoduje na základě dat, stávají se pro něj informací, protože datům přiřazuje nějaký význam a smysl. Data tedy můžeme nazvat potenciálními informacemi. Lidé jsou obklopení zprávami. Některé pro ně důležité zachytí a porozumí jim. Data můžeme ukládat pro pozdější zpracování nebo je převést do jiné podoby, například uložit do počítače nebo napsat na papír a uložit do kartotéky [1].



Obrázek č. 2: Data (Zdroj: převzato z 1, s.5)

2.1.3 Znalosti

Znalosti lze popsat jako informace o tom, jak využít jiné informace a data v různorodých situacích. Rozhodujeme se na základě znalostí, které jsme získali předchozími zkušenostmi. Znalosti získáváme v různých formách a v různé intenzitě každý den. Znalost je informace, které prošla naším uspořádáním a analýzou, abychom ji mohli používat k řešení problémů. Někdy je důležité učinit rozhodnutí rychlé než kvalitní, tedy že čas pro rozhodnutí nesmí překročit čas existence problému [1].

2.2 Databáze

Databáze je soubor vzájemně souvisejících datových položek, které jsou spravovány jako jediná položka. Je to místo, kde se uchovávají data o reálném světě, tak abychom s nimi mohli jednoduše pracovat, mazat a přidávat další. Jednoduchým příkladem může být zápisník učitele, kde uchovává jména jednotlivých žáků a jejich hodnocení.
Tento příklad je však databáze papírová, v dnešní době je to už zastaralý způsob uchovávání informaci. Dnes se již většinou používají databáze relační. Relační databáze je databáze zakládající se na relačním modelu, co je to relační datový model si popíšeme později. Databáze se skládají z entit, které popisujeme pomocí atributů a spojujeme vzájemnými vazbami mezi nimi [2].

Základní datové modely dělíme podle způsobu ukládání dat a druhu vazeb vytvořených mezi nimi [1].

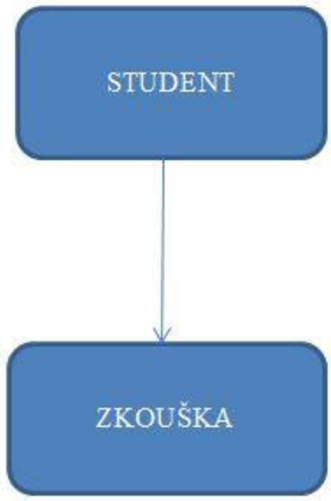
2.2.1 Lineární datový model

Mezi jednotlivými tabulkami nejsou žádné vazby (vztahy). Každá tabulka je brána jako samostatný objekt. Máme dvě tabulky, student a zkouška. Nejsme schopni určit vazby a tím pádem, ani jestli student zkoušku udělal nebo ne. Každá tabulka je brána jako samostatný objekt. Jediné vztahy, které jsme schopni určit, jsou předchůdce a následovník. Příkladem můžou být papírové kartotéky. V současné době se jedná o zastaralý model nevhodný k řešení mého problému [1].

Obrázek č. 3: Lineární datový model (Zdroj: převzato z 1, s.20)

2.2.2 Hierarchický datový model

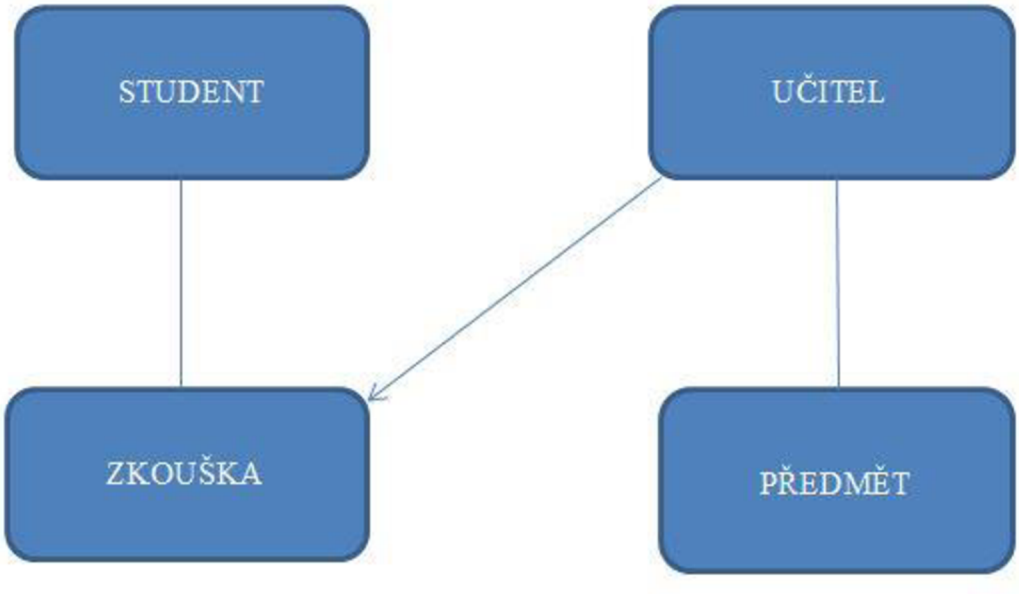
U tohoto modelu již můžeme identifikovat vazby mezi tabulkami. Je to víceúrovňový stromový model, kde se vyskytují tabulky rodičů a tabulky potomků. Rodičovská tabulka může mít více potomků, ale tabulka potomka nemůže mít více rodičů. Jestliže rodičovská tabulka uchovává údaje o studentovi, tak dceřiná tabulka zkouška obsahuje informace o zkoušce. Do tabulky zkouška se lze dostat, pomocí tzv. pointerů, což jsou ukazatele, které vytváří databázový systém, na kterém je model implementován, pouze přes rodičovskou tabulku student. Nikdy ne naopak. Výhodou tohoto modelu je to, že data můžeme získat velmi rychle, protože mezi tabulkami existuje přímé propojení. Nevýhodou je delší doba na reorganizování databáze při vkládání a mazání údajů. I tento model je pro řešení mé bakalářské práce nevhodný [1].



Obrázek č. 4: Hierarchický datový model (Zdroj: převzato z 1, s.21)

2.2.3 Síťový datový model

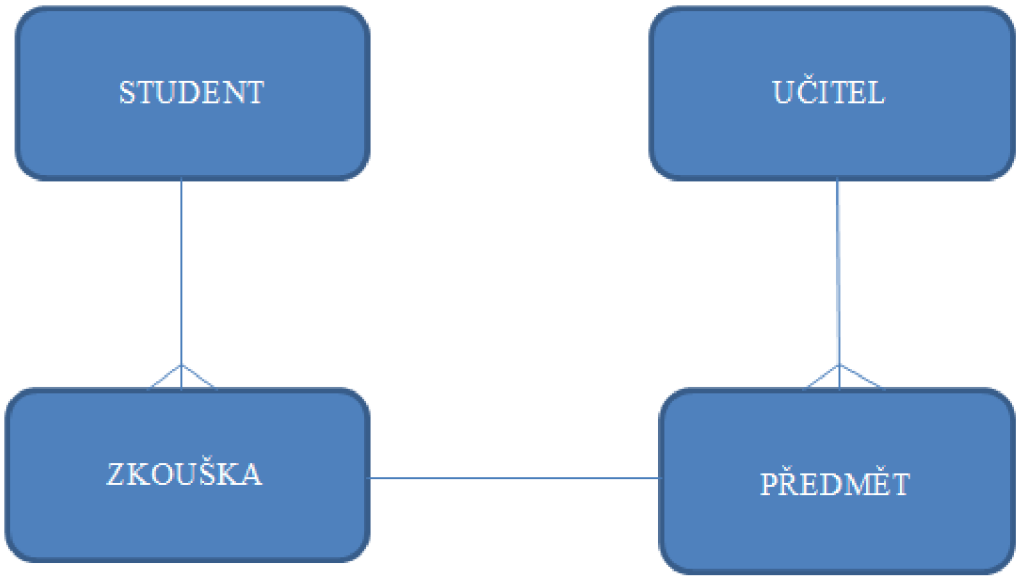
Tento model je velmi podobný hierarchickému datovému modelu. Rozdíl je v tom, že se již nerozlišují tabulky rodičů a potomků. Pointery mohou vést mezi všemi tabulkami databáze různými směry. Výhody a nevýhody jsou stejné jako u předcházejícího modelu. Mezi výhody navíc řadíme libovolné propojení tabulek a tím i urychlení vyhledávání informací [1].



Obrázek č. 5: Síťový datový model (Zdroj: převzato z 1, s.22)

2.2.4 Relační datový model

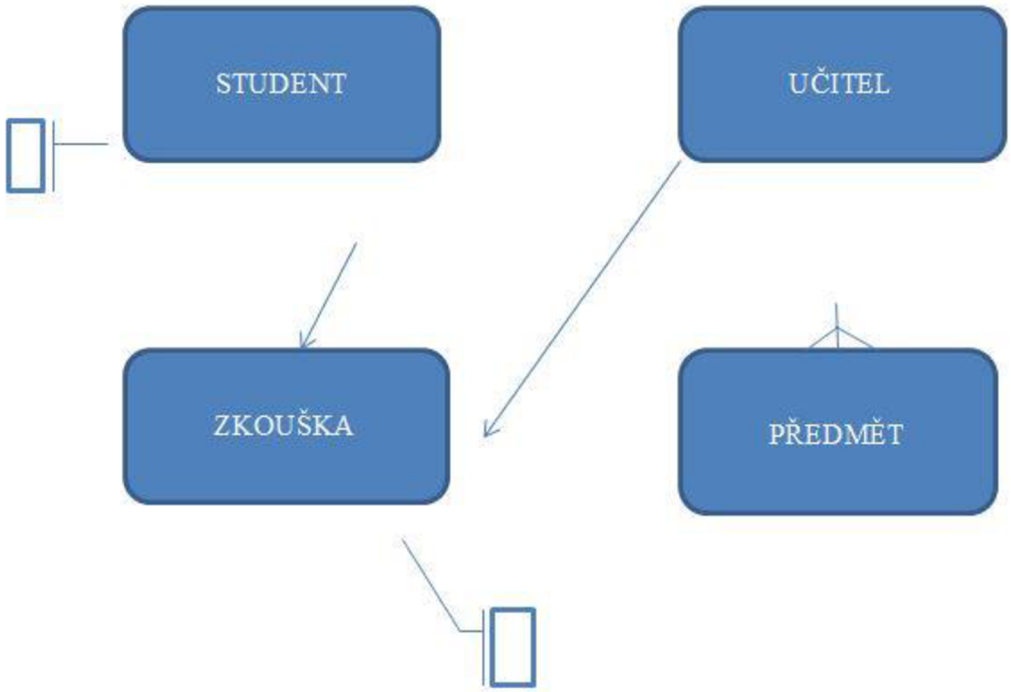
Jedná se o nejvíce rozšířenou a používanou metodu. Vzniká spojením několika lineárních modelů. Data jsou zde ve formě dvourozměrných tabulek. Jednotlivé věty tabulek identifikujeme pomocí jednoznačného primárního klíče. Spojení mezi tabulkami není trvalé, ale pouze po dobu potřebnou k získání dat z tabulek. Tento model jsem zvolil pro řešení mé práce. [1].



Obrázek č. 6: Relační datový model (Zdroj: převzato z 1, s.22)

2.2.5 Objektový datový model

Jedná se o nejmladší datový model. Základním stavebním kamenem těchto datových modelů je objekt (odpovídá pojmu věta). Tento objekt kromě atributů definovány i metody, které určují, jak se objekt chová. Když by byl takovým objektem zkouška studenta tak jeho atributem může být číslo studenta, datum, známka a metodou může být vytvoření hodnocení studenta, kde se kontrolují náležitosti zkoušky jako počet termínů [1].



Obrázek č. 7: Objektový datový model (Zdroj: převzato z 1, s.23)

2.3 Terminologie

V této kapitole si uvedeme některé důležité pojmy z oblasti relačního datového modelování.

2.3.1 Entita

Entita nemusí být jenom fyzický objekt z reálného světa jako je učitel nebo žák, ale může to být i objekt abstraktní jako je předmět nebo zkouška. Tento objekt je schopen vlastní nezávislé existence a je jednoznačně identifikovatelný mezi ostatními entitami [3].

2.3.2 Vazby mezi entitami

Relace mezi tabulkami vyjadřují vztahy mezi objekty z reálného světa, které tyto tabulky představují. Můžeme definovat několik druhů vztahů [3].

1:1 – Jednomu záznamu v jedné tabulce, odpovídá maximálně jeden záznam v tabulce druhé [3].
Přiklad: Jeden student má právě jednu identifikační kartu

1:N – Jednomu záznamu v jedné tabulce, odpovídá více záznamů v tabulce druhé a naopak více záznamů v druhé tabulce odpovídá právě jednomu záznamu v první tabulce [3].
Příklad: Jeden učitel pracuje právě v jedné škole a jedna škola zaměstnává více učitelů.

N:M – Více záznamů v jedné tabulce, odpovídá více záznamům v tabulce druhé a naopak [3].
Příklad: Jeden student navštěvuje více předmětů a jeden předmět navštěvuje více studentů.

2.3.3 Klíče

V relačních databázích se vyskytují tři druhy klíčů.
Primární klíč – Jedná se o jednoznačný identifikátor každého záznamu v tabulce. Musí být jednoznačný tak, že neexistuje v tabulce věta, která má stejnou hodnotu. Musí být minimální. Nemůže obsahovat hodnotu NULL [3].

Cizí klíč – „*Je to sloupec nebo kombinace více sloupců, které jsou propojeny na primární klíč v jiné tabulce*“ [3]. „*Každá hodnota je plně zadaná nebo plně nezadaná*“ [1].

Kandidátní klíč – „*Kandidátní klíč je totéž jako primární klíč, se stejnými vlastnostmi, ale není vybrán jako primární klíč. V relaci může být více kandidátních klíčů, jeden z nich je vybrán jako primární klíč, ostatní kandidátní klíče se nazývají alternativní*“ [1].

2.3.4 Integrita relačního modelu

Modelování dat z reálného světa naráží na omezení teoretického modelu. Integritu chápeme tak, že data v objektu odpovídají reálnému světu. Integritní omezení dělíme na [3] :

Doménová integrita – „*Zajištění, aby každá hodnota atributu byla v souladu s množinou přípustných hodnot*“ [3].

Entitní integrita – „*Zajištění jednoznačné identifikace každého řádku relace – jednoznačný primární klíč*“ [3].

Referenční integrita – „*Cizí klíče (tj. atributy či skupina atributů tvořící v jiné relaci primární klíč) nemohou nabývat hodnoty, které jsou v rozporu s hodnotami odkazovaného primárního klíče – různé způsoby zajištění*“ [3].

2.3.5 Normalizace

Normalizace je proces, který upravuje návrhy datových struktur do podoby, která splňuje normalizační úrovně. Tyto úrovně vycházejí z požadavku na efektivní ukládání dat a minimální redundanci. Původně stačily na pokrytí všech nedostatků první tři formy, ale postupně byly přidány další. Při normalizaci na vyšší formu, by měly být normalizovány všechny nižší úrovně [1].

1. První normální forma

Tabulka je v první normální formě tehdy, jsou-li všechny sloupce atomické (nedělitelné) [3].

2. Druhá normální forma

Tabulka je v druhé normální formě, pokud je v první normální formě a každý atribut vyjma primárního klíče je úplně závislý na celém primárním klíči [3].

3. Třetí normální forma

„*Tabulka je ve třetí normální formě, pokud je ve druhé normální formě a zároveň neexistují závislosti neklíčových sloupců tabulky*“ [3].

4. Boyce – Coddova normální forma

Je to variace třetí normální formy. Tabulka je v Boyce – Coddově normální formě, pokud je ve třetí normální formě a zároveň mezi kandidátními klíči není žádná funkční závislost [3].

5. Čtvrtá normální forma

„*Tabulka je ve čtvrté normální formě tehdy, je-li ve třetí normální formě a popisuje jen jeden fakt anebo souvislost“* [3].

6. Pátá normální forma

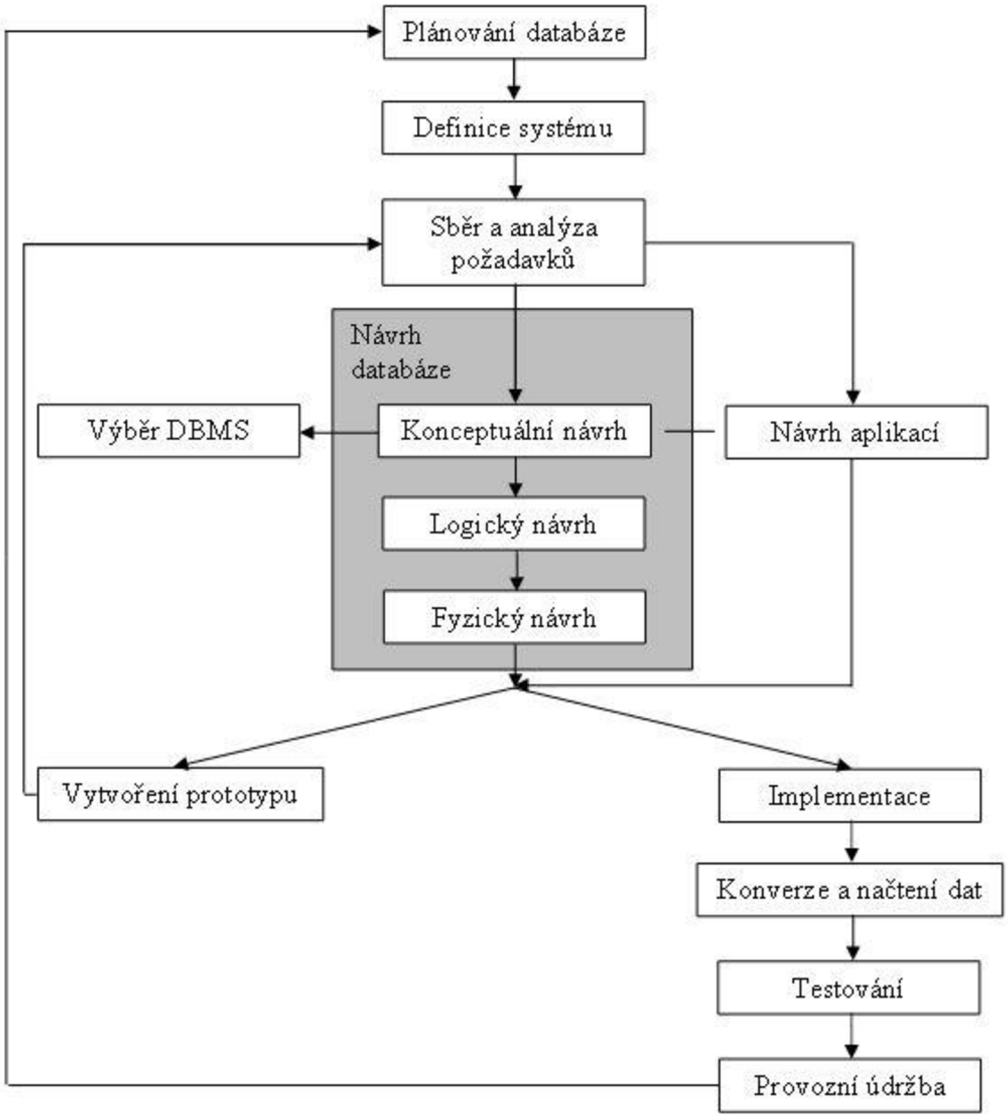
„*Tabulka je v páté normální formě tehdy, je-li ve čtvrté normální formě a není možné do ní přidat nový sloupec, případně skupinu sloupců, aniž by se rozpadla na několik dílčích tabulek“* [3].

2.4 Návrh databáze

Návrh databáze je obtížný proces, protože obsahuje mnoho rozdílných informací. Špatně navrhnutá databáze může neefektivně vyhledávat data nebo data můžou být úplně nedostupná. V této části popíši, jak se má postupovat při návrhu databáze.

2.4.1 Životní cyklus databáze

Životní cyklus databáze začíná v době její potřeby a končí jejím odstraněním. Obrázek č. 8 znázorňuje všechny kroky při vytváření nové databáze. Důležitou částí je návrh databáze, který se skládá z konceptuální, logické a fyzické úrovně. V konceptuální úrovni tvoříme ER diagram, který se stává základem nové databáze. V logické úrovni tento ER diagram přepracujeme, aby splnil podmínky relačních databází. Definujeme primární a cizí klíče, odstraníme vazby N:M. Posledním krokem v návrhové části je fyzická úroveň. Zde tvoříme samotnou databázi pomocí jazyka SQL [6].



Obrázek č. 8: Životní cyklus databáze (Zdroj: 5, s.110)

Pro účely mé práce zmíním a stručně popíši ER diagram a vývojový diagram.

2.4.2 ER diagram

Tento diagram je jedním z nejdůležitějších při návrhu databází. Z ER diagramu vychází samotný SQL kód. Vyjadřuje jednotlivé vazby mezi tabulkami a také typy těchto vazeb [1].

2.4.3 Vývojový diagram

Také jeden z nejpoužívanějších diagramů. Velice jednoduše a přesně zachycuje větvení zpracování dle splnění nebo nesplnění podmínek [1].

2.5 Jazyk SQL a jeho kategorie

Zkratka SQL znamená structured query language, v češtině strukturovaný dotazovací jazyk. SQL byl vyvinut v sedmdesátých letech minulého století v San Jose v Kalifornii v laboratořích IBM [4].

Příkazy jazyka SQL se dělí do několika skupin podle svých funkcí. Jedná se o kategorie DDL, DQL, DML, DCL a příkazy řízení transakcí. Tyto již zmiňované kategorie teď popíši podrobněji [2].

2.5.1DDL (Data Definiton Language)

„Jazyk DDL zahrnuje příkazy SQL, které umožňují uživatelům databáze vytvářet databázové objekty (např. tabulky, pohledy a indexy) a upravovat jejich strukturu. Je důležité si uvědomit, že příkazy jazyka DDL mají vliv na kontejnery, které uchovávají data v databázi, nikoli na vlastní data. Proto existují příkazy DDL pro vytvoření, odstranění a úpravy tabulek, ale žádný z těchto příkazů neumožňuje vytvářet či měnit řádky v těchto tabulkách“ [2].

- CREATE DATABASE – vytvoření databáze
- CREATE TABLE – vytvoření tabulky
- DROP TABLE – smazání tabulky
- ALTER TABLE – úprava tabulky [2]

2.5.2 DQL (Data Query Language)

„Jazyk DQL obsahuje příkazy SQL, které načítají data z databáze. Ačkoli se jedná o velmi důležitou součást jazyka SQL, jsou příkazy jazyka DQL založeny pouze na jediném klíčovém slově: SELECT“ [2].

- SELECT – výběr dat [2]

2.5.3 DML (Data Manipulation Language)

„Součástí jazyka DML jsou příkazy SQL, které umožňují uživatelům přidávat data do databáze (v podobě řádků nebo tabulek), odebírat data z databáze a měnit stávající data“ [2].

- INSERT – vložení dat
- UPDATE – úprava dat
- DELETE – smazání dat [2]

2.5.4 DCL (Data Control Language)

„Do jazyka DCL patří příkazy SQL, které správcům dovolují řídit přístup k datům v databázi a používat různá systémová oprávnění SŘBD, jako je například funkce pro spuštění nebo vypnutí databáze“ [2].

- GRANT – přidělení privilegia
- CREATE USER – vytvoření uživatele
- DROP USER – smazání uživatele
- ALTER USER – úprava uživatele [2]

2.5.5 Příkazy řízení transakcí

„Databázová transakce je sada příkazů, kterou databázový uživatel požaduje zpracovat jako nedělitelnou jednotku. To znamená, že transakce musí být kompletně úspěšná nebo neúspěšná. Příkazy řídící databázové transakce přesně neodpovídají syntaxi příkazů jazyka SQL, ale mají značný vliv na chování těch příkazů SQL, které jsou součástí transakcí“ [2].

- BEGIN TRANSACTION – uvození transakce
- COMMIT TRANSACTION – úspěšné ukončení transakce
- ROLLBACK TRANSACTION – neúspěšné ukončení transakce [2]

2.6 Datové typy jazyka SQL

„Datové typy jsou měřítkem atributů“ [3].

2.6.1 Číselné datové typy

Slouží k ukládání číselných údajů, je vždy definován rozsah [3].

- DECIMAL – rozsah je od $-10^{38} + 1$ do $10^{38} - 1$
- FLOAT – s pohyblivou desetinnou čárkou a rozsahem od $-1{,}79^{308}$ až $-2{,}33^{308}$, 0, $2{,}23^{308}$ až $1{,}79^{308}$
- DOUBLE – s pohyblivou desetinnou čárkou a dvojnásobnou přesností
- REAL – s pohyblivou desetinnou čárkou a rozsahem od $-3{,}4^{38}$ až $-1{,}18^{38}$, 0, $1{,}18^{38}$ až $3{,}4^{38}$ [3]

2.6.2 Datové typy pro vyjádření finančních částek

„Do skupiny číselných datových typů patří i datové typy pro vyjádření finanční částky v peněžní měně. Finanční částky se vyjadřují a počítají na pevný počet desetinných míst, zpravidla na dvě nebo čtyři desetinná místa“ [3].

- MONEY – rozsah je od -2^{63} do 2^{63} -1
- SMALLMONEY – rozsah je od -214 748,3648 do 214 748,3647 [3]

2.6.3 Celočíselné datové typy

- BIT – slouží k vyjádření dvou hodnot, 0 nebo 1 anebo pravda/nepravda
- INT – celé číslo v rozsahu od -2^{31} do 2^{31} -1, zabírá 4 bajty
- SMALLINT – celé číslo v rozsahu -2^{15} do 2^{15} -1, zabírá 2 bajty
- TINYINT – celé číslo v rozsahu od 0 do 255, zabírá 1 bajt [3]

2.6.4 Znakové datové typy

- CHAR() – slouží k uložení textového řetězce pevné délky, podle parametru v závorce
- VARCHAR() – slouží k uložení textového řetězce proměnné délky, podle parametru v závorce [3]

2.6.5 Datové typy pro datum a čas

- DATETIME – ukládá data jako jedinou hodnotu ve formátu 01/01/1753 až 12/31/9999
- SMALLDATETIME – rozsah 01/01/1900 až 12/31/2079
- GETDATE – vloží aktuální datum a čas [3]

3. Analýza současného stavu

V této části se seznámíme s analyzovaným problémem a současnou situací školy. Poté popíši hardwarové i softwarové vybavení školy a závěrem se seznámíme se školními informačními systémy, které škola používá.

3.1 Základní údaje o škole

- **Název**: Střední průmyslová škola, Obchodní akademie a Jazyková škola s právem státní jazykové zkoušky, Frýdek-Místek, příspěvková organizace
- **Sídlo**: 28. října 1598, 738 01 Frýdek-Místek
- **IČO**: 00601381

3.2 Historie školy a její vývoj do dnešní podoby

Počátek Střední průmyslové školy je třeba hledat již v roce 1955, kdy bylo rozhodnuto o tom, že je třeba zřídit v místecké části okresního města Frýdek-Místek novou střední školu, která by produkovala absolventy s technickými znalostmi a uspokojovala potřeby průmyslových firem na Ostravsku a v jeho okolí. 1. Září 1958 se pro 1. ročník otevřely první dva obory, hutnictví a strojírenství. V roce 1961 měla škola už všechny 4 ročníky těchto dvou oborů. V roce 1973 přibyl obor slévárenství. Po dlouhou dobu zůstal tento stav nezměněn, až do roku 1988, kdy se otevřel nový obor textilní technologie. V roce 1990 kulminoval stav studentů na počtu 902. Škola spolupracovala po celou dobu s podniky jako Vítkovické strojírny a železárny, Slezan Frýdek-Místek a Ferrum. Po roce 1991 vzrostl zájem o administrativní obory a tak vznikly úplně nové studijní programy jako strojírenská a hutnická administrativa. Po roce 2000 se studijní programy opět měnily a to tak, že přibyl stavební obor, technické zařízení budov a také přibylo technické lyceum, které primárně připravovalo žáky na studium technických vysokých škol. Škola začala spolupracovat se společnostmi Arcelor Mittal Ostrava a.s. a s Třineckými železárnami a ocelárnami a.s., které poskytovaly finanční podporu nejlepším studentům a těm, kteří dojížděli z okolí. V roce 2000 škola získala mezinárodní certifikáty Autodesk Academia Certificate of Completion a Microsoft certifikát, který škole umožnil zřídit akreditované středisko výpočetní techniky. V roce 2004 škola získala statut informační centra SIPVZ (státní informační politika ve vzdělání). V roce 2006 škola získala dotaci 3,6 mil. Kč na nákup nových CNC strojů. Programovat tyto stroje se učí nejen studenti středních škol, ale i studenti nedaleké

Vysoké Školy Báňské. Žáci této školy se pravidelně umísťují na předních příčkách v celostátních a mezinárodních průmyslových soutěžích. Mezi slavné absolventy této školy patří mj. horolezec Libor Uher [7].

3.3 Současný stav

V dnešní době jsou aktuální informace o studentech uchovávány elektronicky pomocí informačního systému Bakaláři. V okamžiku ukončení studia se informace ze systému Bakaláři převedou do papírové podoby a jsou uloženy do školního archivu. Školní archiv se nachází ve sklepních prostorách budovy, ve které škola sídlí. Ve školním archivu se nachází maturitní protokoly a žákovské práce. Maturitní protokoly obsahují údaje o jednotlivých studentech, jejich hodnocení z maturitních předmětů a údaje o tom, kdo je hodnotil. Žákovské práce jsou různé čtvrtletní a pololetní písemné práce a také různé předepsané školní projekty, které se archivují po dobu pěti let. Tyto údaje se třídí do svazků podle předmětů a školního roku. Každý svazek obsahuje předepsané písemné práce z předmětů za třídu.

Škola si uchovává předepsané dokumenty dle zákona o archivnictví č. 499/2004 Sb. o archivnictví a spisové službě a jeho prováděcí vyhlášky č. 646/2004 Sb. o podrobnostech výkonu spisové služby, dále ze zákona o účetnictví č. 563/91 Sb. a jeho pozdějších novel a směrnice MŠ ČSR ze dne 13.6.1986 č.j. 18200/86-49, který byl vydán jako skartační řád pro školy a školská zařízení [8].
Tyto dokumenty se musí archivovat různou dobu podle důležitosti informace. Všechny práce jsou archivovány dle Skartačního a spisového řádu a jsou zařazeny do skupin, podle skartačních znaků A a S. S – stoupa případně skart, po uplynutí skartační lhůty se zničí. A – po lhůtě se předává do archivu k trvalému uložení. Práce jsou rozděleny na písemné práce žáků, které jsou zařazeny do skupiny S5, a maturitní zkoušku, které jsou zařazeny do skupiny A45. Maturitní zkouška je dále rozdělena na písemnou práci a protokol o maturitní zkoušce. Písemné žákovské práce se archivují 5 let, maturitní protokoly 45 let. Poté se tyto maturitní práce předají místní pobočce matriky. Vzhledem k tomu, že jde o školu se 650 studenty, tak tyto informace o všech studentech, kteří prošli školou za posledních 45 let, zabírají obrovské množství místa a také hledání relevantních informací v těchto archívech zabere hodně času, který by mohl být využit lépe.

3.4 Informační technologie školy

Střední průmyslová škola ve Frýdku-Místku má několik specializovaných učeben, kde jsou zapotřebí informační technologie.

Jedná se o učebny výpočetní techniky, kterých je ve škole 7. V každé z těchto učeben se nachází dataprojektor, který promítá obraz na plátno. Tyto projektory jsou připojeny ke stolnímu počítači, ke kterému má přístup pouze učitel. Pro žáky jsou v těchto učebnách přístupné průměrně výkonné počítačové sestavy. Vedení školy se snaží průběžně doplňovat a renovovat hardware, tak aby mohly bezproblémově fungovat všechny programy, se kterými studenti během svého studia přijdou do styku. Počítačové učebny se po skončení výuky zamykají a studenti proto nemají do těchto učeben volný přístup.

Dále mezi učebny, kde se nachází výpočetní technika, můžeme zařadit různé laboratoře a měřící místnosti. V těchto učebnách žáci přicházejí do styku se specializovaným softwarem a hardwarem. Každá laboratoř je vybavena stroji a potřebným zařízením pro výuku žáků. Většina strojů a zařízení je však již velice zastaralá a neodpovídají tedy současnému technologickému vývoji a ani potřebám zaměstnavatelů na vzdělané absolventy se zkušenostmi s obsluhou strojů a zařízení. Škola se proto nedávno zapojila do projektu financovaného Evropskou unií s cílem modernizovat tyto laboratoře. Díky realizaci projektu budou laboratoře zmodernizovány, čímž dojde ke zkvalitnění a rozšíření výuky - nové vybavení laboratoří umožní mimo oblast konstruování a obrábění pomocí CNC strojů posuzovat i kvalitu vyrobené součásti. V současné době je výuka některých tematických celků vyučována převážně v teoretické rovině. Pořízené přístrojové vybavení umožní doplnit výuku praktickými ukázkami a prováděním vlastního měření.

V budově školy se nachází také přednášková aula s kapacitou 100 míst, ve které je umístěn projektor a plátno k promítání. V budově školy je dostupný wifi signál.

3.5 Školní informační systémy

V dnešní době jsou školní informační systémy nepostradatelnou součástí školní agendy. Školní informační systém umožňuje komunikaci a zpracování informací. Zajišťují ukládání informací důležitých pro činnost školy a její komunikaci uvnitř i vně školy. Informační systém by měl vytvářet prostředí pro efektivnější činnost managementu školy.

Školní informační systémy prošly za poslední dvě desetiletí určitým vývojem, od papírových kartoték až po komplexní informační systémy. Hlavní zaměření na administrativu spojenou s pedagogickým procesem zůstává. Běžné jsou funkce jako tisk vysvědčení, tvorba rozvrhů nebo suplování. Vývoj však jde dál, a aby producenti drželi krok s konkurencí, musí svou nabídku neustále rozšiřovat a zkvalitňovat. Důsledkem je trend, kdy jsou systémy primárně zaměřené na evidenci žáků a přípravu rozvrhu rozšiřovány o nové moduly, pro jejichž zpracování byly v minulých letech používány jednoúčelové aplikace, např. správa knihovny a výpůjček, hospodaření školní jídelny apod. Výběr vhodného informačního systému bývá ovlivněn několika kritérii:

- Zázemí a renomé výrobce – Stabilní společnost s tradicí, která školní IS dodává do škol již 10 let, bude získávat nové zakázky s větší pravděpodobností, než firma, která vznikla loni.

- Podpora uživatelů – Dostupnost servisních technických pracovníků v lokalitě školy je důležitá. Kvalitní web s propracovaným systémem zveřejněných často kladených otázek a odpovědí potom zase znakem profesionality.

- Komplexnost – Výhoda je velká nabídka doplňkových modulů, dle výběru školy.

- Možnost exportu a importu dat, přístup přes internet a také cena IS [9].

3.5.1 Moodle

Moodle je zkratka Modular Object-Oriented Dynamic Learning Environment, v češtině Modulární objektově orientované dynamické prostředí pro výuku. Je to softwarový balíček pro tvorbu výukových systémů a elektronických kurzů na internetu. Moodle je poskytován zdarma jako Open Source pod veřejnou licencí GNU. Moodle lze použít na jakémkoliv počítači s fungujícím php [10].

3.5.2 Bakaláři

Je to školní informační systém, který nabízí firma Bakaláři software s.r.o. Program Bakaláři řeší evidenci žáků a zaměstnanců, klasifikaci, 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. Program je vytvořen přímo pro prostředí českých škol a je kompatibilní s MS SQL Server [11].

3.6 Zdůvodnění návrhu

Žádný z těchto informačních systémů, které škola využívá, nenabízí elektronickou archivaci absolventů školy včetně jejich maturitních protokolů a žákovských prací. Po ukončení studia jsou data z IS Bakaláři převedena do papírové podoby a uložena ve školním archivu. Často se stává, že po několika letech se absolventi vracejí do školy s tím, že ztratili své maturitní vysvědčení a potřebují doložit z různých důvodů své vzdělání. Skrze svůj maturitní protokol, který je uložen právě ve školním archivu mají tito lidé jedinou šanci dostat své původní maturitní vysvědčení. S těmito prosbami se obracejí na správce archivu, který má tuto agendu na starosti. Údržba tohoto archivu, který je ve formě papírové kartotéky, je poměrně náročnou časovou záležitostí. Musejí se přidávat nová data, skartovat stará, zakládat nové záložky atd. Nemluvně o velkém využití papíru. Hledání relevantních informací v této kartotéce zabere také určité množství času a je neefektivní. Proto jsem se rozhodl navrhnout řešení tohoto problému.

Databáze bude fungovat zcela samostatně. Využiji zde kompatibility IS Bakaláři s MS SQL Serverem. Údaje o studentech a zaměstnancích (jméno, příjmení, rodné číslo atd..) budou převedeny z IS Bakaláři do mnou vytvořené databáze, tak ať učitelé nemusejí dělat dvojí práci se vkládáním studentů a učitelů. Poté, co se ze studentů stanou absolventi, pověřená osoba zapíše jejich hodnocení do databáze, včetně data, kdy došlo k zápisu. Databáze bude automaticky hlídat datum, kdy vyprší povinná minimální doba archivace.

4. Vlastní návrh řešení

V této části budu navrhovat řešení analyzovaného problému. Nejprve vytvořím konceptuální návrh, identifikuji všechny entity, které budu v databázi používat a poté vytvořím ER diagram. Dále vytvořím logický návrh databáze, zde znázorním všechny tabulky a jejich klíče. Jako poslední krok bude fyzický návrh databáze, kde se jedná o samotné vytvoření tabulek, procedur a spouští pomocí jazyka SQL. Databázi budu tvořit v prostředí MS SQL Server 2014.

4.1 Požadavky na databázi

Požadavky na vypracování databáze jsem konzultoval se správcem školního archivu. Jedním ze stěžejních požadavků bylo navrhnout databázi, která by mohla nahradit a zpřehlednit školní archiv, který v současné době uchovává data v papírové podobě. Databáze bude umět uchovávat data o aktuálních studentech a poté, co studenti ukončí své studium, změní se jim statut studenta na statut absolventa.

4.2 Uživatelé databáze

K databázi bude přistupovat několik skupin uživatelů. Těmto skupinám budou přiřazeny různá práva k práci s touto databází. Jedná se o administrátora databáze, učitele, studenty, zákonné zástupce studentů a nakonec absolventy školy.

4.2.1 Administrátor

Administrátor bude mít nejvyšší přístupová práva k manipulaci s databází. Bude mít na starosti provoz databáze a bude řešit případné problémy. Má na starosti instalaci, údržbu a upgrade serveru. Má dohled nad serverem z bezpečnostního hlediska a z hlediska zátěže serveru a databáze samotné. O integritu dat se budou starat především databázové spouště neboli triggery.

4.2.2 Učitelé

Na Střední průmyslové školy v současné době vyučuje 63 učitelů, kteří mohou přistupovat do databáze a zapisovat nové údaje. Všichni učitelé ovládají počítač na takové úrovni, která jim práci s databází umožní. S databází bude nejčastěji pracovat správce archivu. Změna papírového archivu na elektronickou databázi je vítaná z důvodu uspoření času, zjednodušení práce s velkým množstvím dat a možnosti práce

s více daty současně. Učitelé budou mít právo vkládat nové údaje do databáze, upravovat současná data a také tyto data číst.

4.2.3 Žáci a jejich zákonní zástupci

V současné době navštěvuje školu 646 žáků. Tyto skupiny uživatelů do databáze nebudou moci zasahovat, vkládat nová a měnit stávající data. Žáci a jejich zákonní zástupci budou moci data pouze prohlížet. Poslední skupinou jsou absolventi školy, budou mít stejná práva jako současní žáci, ale pro pořádek budou mít udělen vlastní statut.

4.3 Konceptuální návrh

Cílem konceptuálního návrhu je vytvoření ER diagramu, který bude splňovat požadavky na databázi. Nejprve však musíme identifikovat všechny entity a relace v databázi.

4.3.1 Identifikace entit

V tomto kroku identifikujeme všechny entity, které se v databázi nachází. Stručně je popíši a zjistím přibližný počet výskytů.

Název entity	Popis entity	Počet výskytů
Třída	Seznam tříd	26
Ročník	Školní rok	1 za rok
Zástupce	Seznam zákonných zástupců studentů	asi 1000
Oprávnění	Oprávnění uživatelů	5
Absolvent	Seznam absolventů	asi 150 za rok
Písemná	Hodnocení písemných maturitních prací	asi 150 za rok
Praktická	Hodnocení praktických maturitních prací	asi 150 za rok
Učitel	Seznam učitelů	asi 60
Absolventské práce	Hodnocení absolventských prací	asi 150 za rok
Zástupce studenta	Přiřazení zástupců ke studentům	asi 1000
Maturita	Hodnocení ústní maturitní zkoušky	asi 150 za rok
Kontakt učitele	Kontakt na učitele	asi 100
Student	Seznam studentů	asi 650
Kontakt studenta	Kontakt na studenta	asi 1000
Předmět	Seznam vyučovaných předmětů	asi 20
Žákovské práce	Hodnocení žákovských prací	asi 150 za rok
Vyučující	Přiřazení učitelů k předmětům	asi 150

Tabulka č.1: Identifikace entit (Zdroj: Vlastní zpracování)

4.3.2 Identifikace relací

V tomto kroku identifikujeme relace mezi entitami, určíme kardinalitu vztahu a popis vztahu.

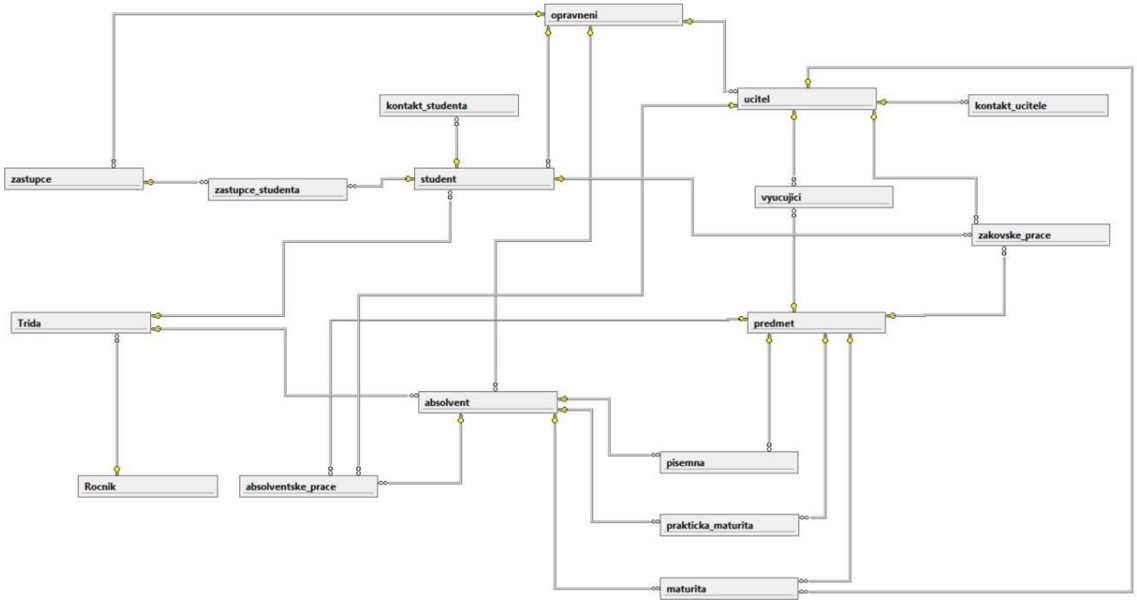
Entity	Relace mezi entitami	Popis vztahu
Třída-Absolvent	1:N	Kolik má třída absolventů? N. Jeden absolvent chodil do kolika tříd? 1.
Ročník-Třída	1:N	Kolik tříd má ročník? N. Jedna třída náleží do kolika ročníků? 1.
Třída-Student	1:N	Kolik studentů navštěvuje třídu? N. Jeden student navštěvuje kolik tříd? 1.
Absolvent-Písemná	1:N	Kolik absolventů má písemnou maturitu? N. Jedna písemná maturita náleží kolika absolventům? 1.
Absolvent-Maturita	1:N	Kolik absolventů má ústní maturitu? N. Jedna maturita náleží kolika studentům? 1.
Absolvent-Absolventská práce	1:N	Kolik absolventů má absolventskou práci? N. Jedna práce náleží kolika absolventům? 1.
Absolvent-Praktická maturita	1:N	Kolik absolventů má praktickou maturitu? N. Jedna praktická maturita náleží kolika absolventům? 1.
Oprávnění-Absolvent	1:N	Kolik absolventů má nějaké oprávnění? N. Jedno oprávnění náleží kolika absolventům? 1.
Oprávnění-Zástupce	1:N	Kolik zástupců má oprávnění? N. Jedno oprávnění náleží kolika zástupcům? 1.
Oprávnění-Student	1:N	Kolik studentů má oprávnění? N. Jedno oprávnění náleží kolika studentům? 1.
Oprávnění-Učitel	1:N	Kolik učitelů má oprávnění? N. Jedno oprávnění náleží kolika učitelům? 1.
Zástupce-Student	N:M	Kolik studentů má svého zástupce? N. Jeden zástupce může mít kolik studentů? M
Učitel-Absolventská práce	1:N	Kolik absolventských prací hodnotil učitel? N. Jedna absolventská práce byla hodnocena kolika učiteli? 1.

Učitel-Maturita	1:N	Kolik maturit hodnotil učitel? N. Jedna maturita byla hodnocena kolika učiteli? 1.
Učitel-Předmět	N:M	Kolik předmětů může učit jeden učitel? N. Jeden předmět může být vyučován kolika učiteli? M.
Učitel-Žákovská práce	1:N	Kolik žákovských prací může být hodnoceno jedním učitelem? N. Jedna žákovská práce je hodnocena kolika učiteli? 1.
Učitel-Kontakt učitele	1:N	Kolik kontaktů může mít učitel? N. Jeden kontakt patří kolika učitelům? 1.
Student-Kontakt Studenta	1:N	Kolik kontaktů může mít student? N. Jeden kontakt patří kolika studentům? 1.
Student-Žákovská práce	1:N	Kolik žákovských prací může být napsáno studentem? N. Jedna práce patří kolika studentům? 1.
Předmět-Absolventská práce	1:N	Kolik prací může být v jednom předmětu? N. Jedna práce náleží kolika předmětům? 1.
Předmět-Maturita	1:N	Kolik maturitních zkoušek může být v jenom předmětu? N. Jedna maturitní zkouška náleží kolika předmětům? 1.
Předmět-Žákovské práce	1:N	Kolik žákovských prací může být v jenom předmětu? N. Jedna práce náleží kolika předmětům? 1.
Předmět-Písemná	1:N	Kolik písemných maturitních prací může být v jednom předmětu? N. Jedna písemná maturitní zkouška náleží kolika předmětům? 1.
Předmět-Praktická maturita	1:N	Kolik praktických maturitních zkoušek může být v jednom předmětu? N. Jedna praktická maturitní zkouška náleží kolika předmětům? 1.

Tabulka č. 2: Identifikace relací (Zdroj: Vlastní zpracování)

Jak je v tabulce č.2 vidět, nachází se zde dvě vazby N:M. Jedná se o entity Zástupce – Student a Učitel – Předmět. Musím proto provést dekompozici vztahu. U entit Zástupce – Student vytvořím novou tabulku Zástupce studenta a vložím do ní primární klíče obou entit. Tímto obě tabulky dekomponuji ze vztahu N:M na 1:N a N:1. Obdobně budu

postupovat s entitami Učitel – Předmět, kdy jsem vytvořil novou tabulku Vyučující. Výsledný ER diagram popisující pouze entity a vazby mezi nimi vypadá následovně:



Obrázek č. 9: ER diagram – Entity (Zdroj: Vlastní zpracování)

4.4 Logický návrh

V logickém návrhu se již tvoří samotné tabulky databáze. Dochází zde ke kontrole tabulek s využitím normalizace, kontrole na podporu uživatelských transakcí a posouzení návrhu s budoucími uživateli databáze, které zastupuje správce školního archivu.

4.4.1 Datový slovník

Datový slovník je jedním z důležitých dokumentů potřebných pro tvorbu databáze. Jsou v něm obsaženy všechny entity a dá se z něj vyčíst, do jakých schémat jsme tyto entity rozdělili. Každá tabulka je podrobně popsána. Vidíme zde všechny položky, typ položky, délku položky, primární a cizí klíče a také jestli má položka nějaké omezení. Při tvorbě SQL databáze se postupuje právě podle této tabulky.

Tabulka	Položka	Typ	Délka	Klíč	Omezení
Oprávnění	ID_opravneni	int		PK	not null
	druh	varchar	20		not null
	popis	varchar	100		
Ročník	ID_rok	int		PK	not null
	Skolni_rok	varchar	9		not null
Třída	ID_trida	int		PK	not null
	ID_rok	int		FK	not null
	zkratka	varchar	6		not null

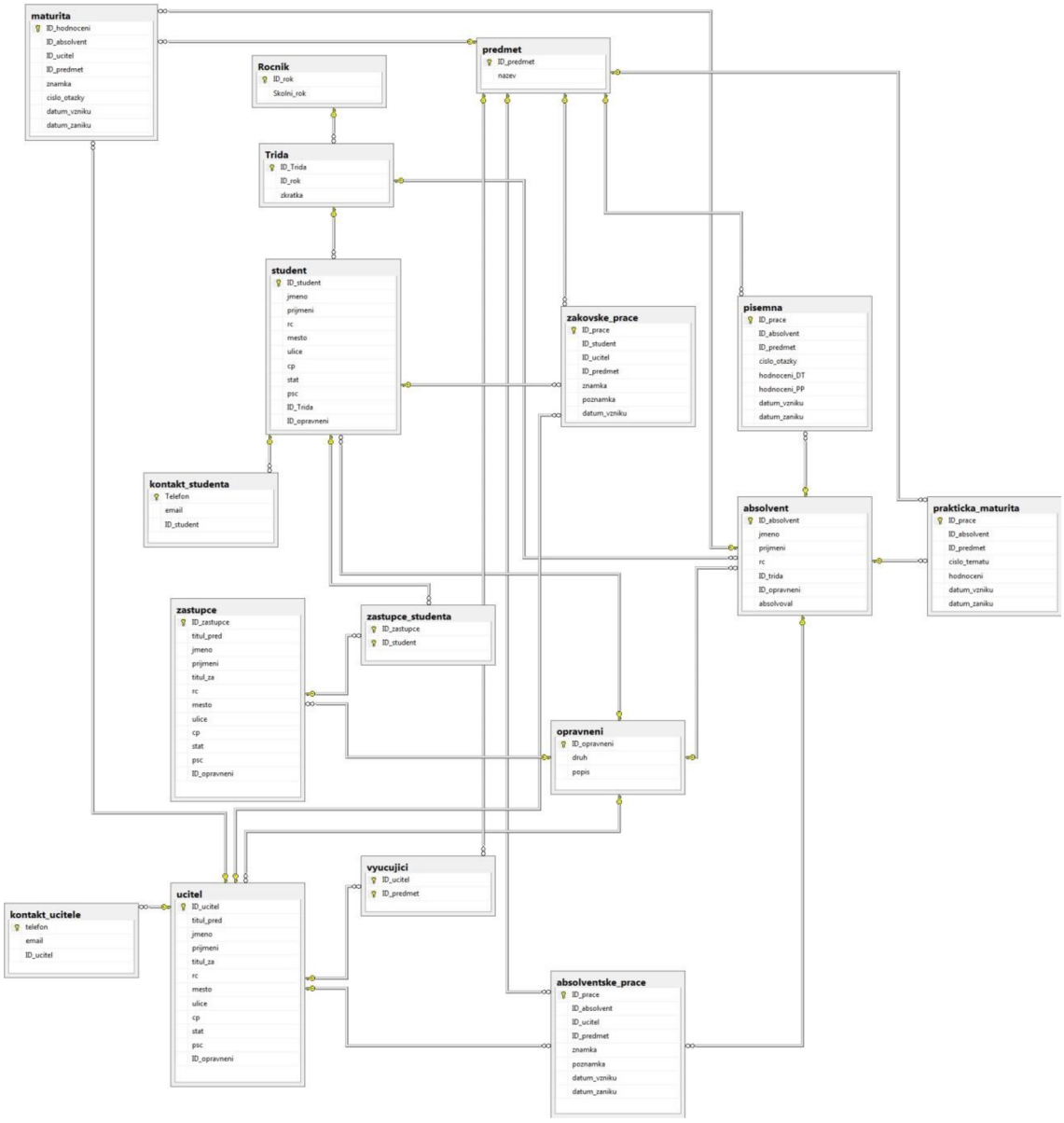
Student	ID_student	int		PK	not null
	jmeno	varchar	20		not null
	prijmeni	varchar	30		not null
	rc	char	10		not null
	město	varchar	30		not null
	ulice	varchar	25		not null
	cp	varchar	10		not null
	stat	varchar	20		not null
	psc	char	5		not null
	ID_Trida	int		FK	not null
	ID_opravneni	int		FK	not null
Učitel	ID_ucitel	int		PK	not null
	titul_pred	varchar	10		
	jmeno	varchar	20		not null
	prijmeni	varchar	30		not null
	titul_za	varchar	10		
	rc	char	10		not null
	město	varchar	30		not null
	ulice	varchar	25		not null
	cp	varchar	10		not null
	stat	varchar	20		not null
	psc	char	5		not null
	ID_opravneni	int			not null
Předmět	ID_predmet	int		PK	not null
	nazev	varchar	20		not null
Vyučující	ID_ucitel	int		PK,FK	not null
	ID_predmet	int		PK,FK	not null
Zástupce	ID_zastupce	int		PK	not null
	titul_pred	varchar	10		
	jmeno	varchar	20		not null
	prijmeni	varchar	30		not null
	titul_za	varchar	10		
	rc	char	10		not null
	město	varchar	30		not null
	ulice	varchar	25		not null
	cp	varchar	10		not null
	stat	varchar	20		not null
	psc	char	5		not null
	ID_opravneni	int			not null
Zástupce studenta	ID_zastupce	int		PK,FK	not null
	ID_student	int		PK,FK	not null
Kontakt studenta	telefon	char	9	PK	not null
	email	varchar	40		
	ID_student	int		FK	not null
Kontakt učitele	telefon	char	9	PK	not null

	email	varchar	40		
	ID_ucitel	int		FK	not null
Absolvent	ID_absolvent	int		PK	not null
	jmeno	varchar	20		not null
	prijmeni	varchar	30		not null
	rc	char	10		not null
	ID_trida	int		FK	not null
	ID_opravneni	int		FK	not null
	absolvoval	date			not null
Maturita	ID_hodnoceni	int		PK	not null
	ID_absolvent	int		FK	not null
	ID_ucitel	int		FK	not null
	ID_predmet	int		FK	not null
	znamka	int			not null
	cislo_otazky	int			not null
	datum_vzniku	date			not null
	datum_zaniku	date			
Žákovské práce	ID_prace	int		PK	not null
	ID_student	int		FK	not null
	ID_ucitel	int		FK	not null
	ID_predmet	int		FK	not null
	znamka	int			not null
	poznamka	varchar	500		
	datum_vzniku	date			not null
Absolventské práce	ID_prace	int		PK	not null
	ID_absolvent	int		FK	not null
	ID_ucitel	int		FK	not null
	ID_predmet	int		FK	not null
	znamka	int			not null
	poznamka	varchar	500		
	datum_vzniku	date			not null
	datum_zaniku	date			
Písemná	ID_prace	int		PK	not null
	ID_absolvent	int		FK	not null
	ID_predmet	int		FK	not null
	cislo_otazky	int			not null
	hodnoceni_DT	int			
	hodnoceni_PP	int			
	datum_vzniku	date			not null
	datum_zaniku	date			
Praktická maturita	ID_prace	int		PK	not null
	ID_absolvent	int		FK	not null
	ID_predmet	int		FK	not null
	cislo_tematu	int			not null
	hodnoceni	int			not null

	datum_vzniku	date	not null
	datum_zaniku	date	

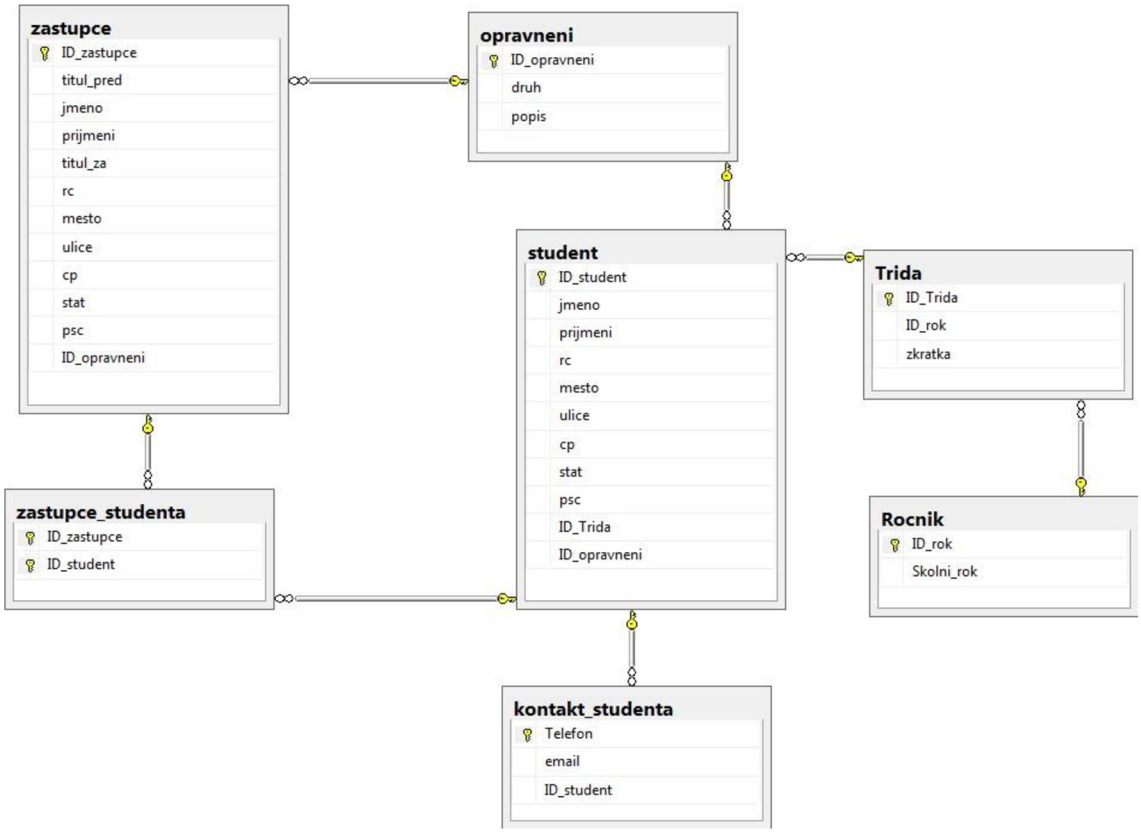
Tabulka č. 3: Datový slovník (Zdroj: Vlastní zpracování)

Nyní máme vše potřebné pro vytvoření kompletního ER diagramu. Jeho finální podoba včetně všech atributů je zobrazena na obrázku č. 10. Poté diagram rozdělím do jednotlivých částí podle jeho funkčnosti a vysvětlím, jaké v něm probíhají procesy.



Obrázek č. 10: Finální ER diagram (Zdroj: Vlastní zpracování)

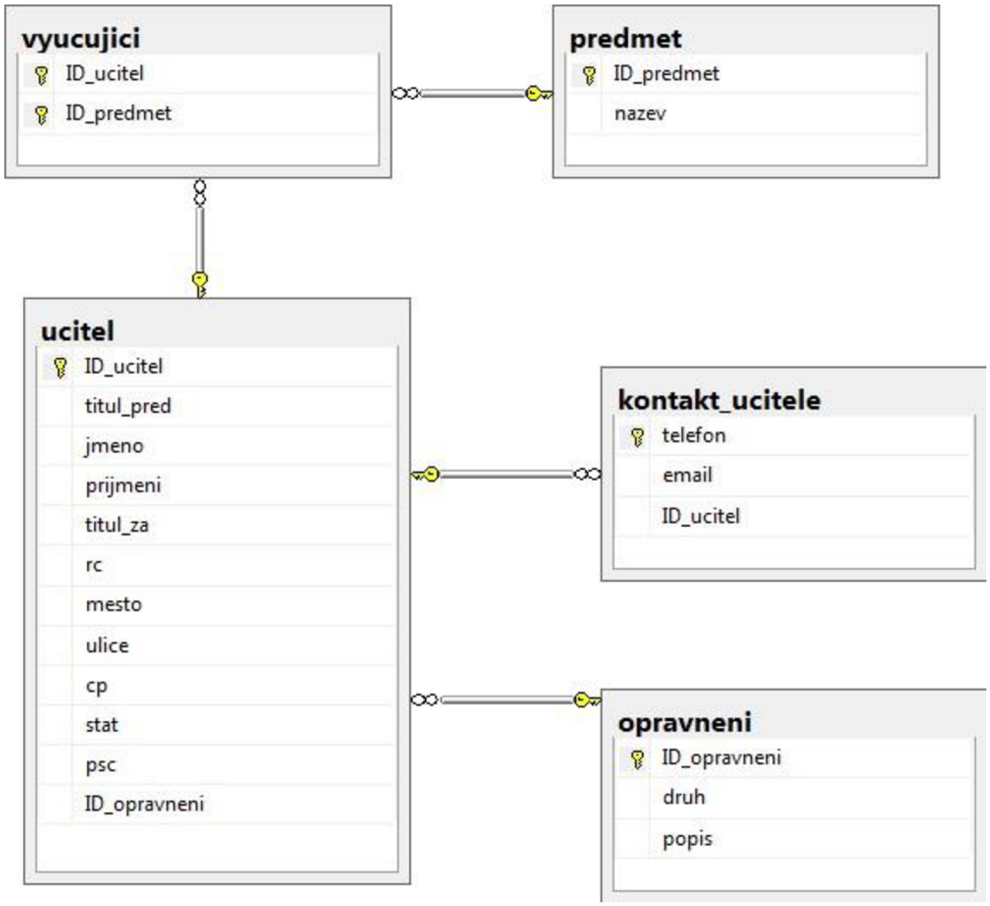
4.4.2 Evidence studentů



Obrázek č. 11: Evidence studentů (Zdroj: Vlastní zpracování)

Primárním klíčem tabulky student jsem zvolil jeho ID, které se bude automaticky generovat při přidání nového studenta. Studentské ID bude mít student stejné po celou dobu studia a toto ID si student ponechá i po skončení studia, později zmíním z jakého důvodu. V tabulce zástupce jsem zvolil obdobný postup. Primárním klíčem je zde ID Zástupce. Studenty a zástupce jsem propojil přes tabulku zástupce studenta, protože každý student může mít více zástupců, např. matku a otce. Tabulku oprávnění tvoří číselník s různými druhy oprávnění pro různé druhy skupin, které s databází budou pracovat. Kontakty na studenty jsem se rozhodl uložit do samostatné tabulky a to z důvodu, že každý student může uvést kontaktů více. Student je zařazen do určené třídy a třída je zařazena do školního roku.

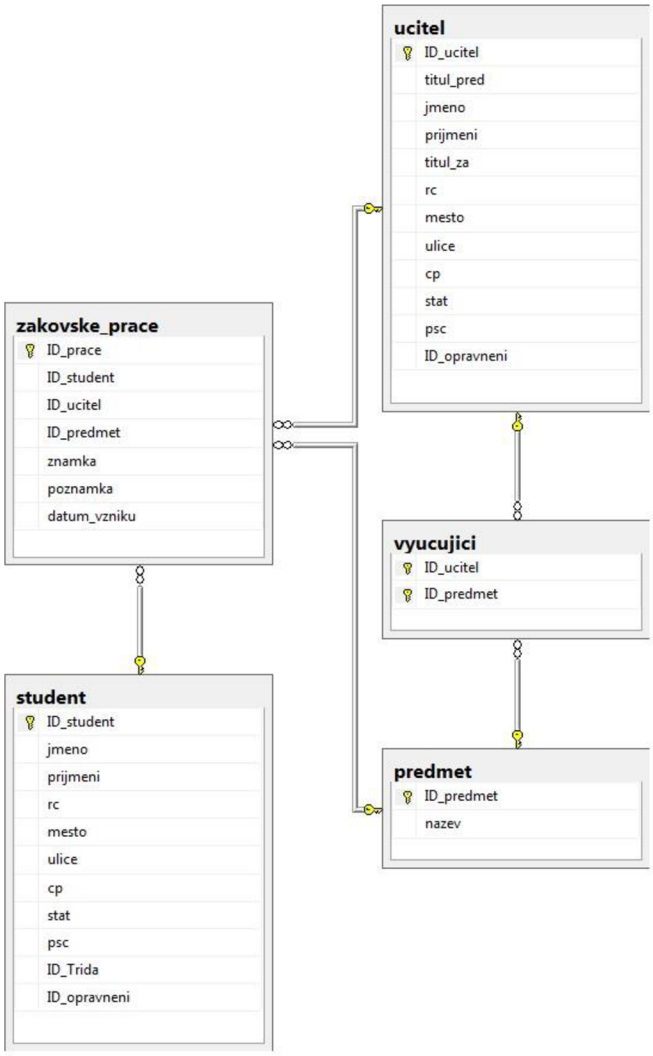
4.4.3 Evidence učitelů



Obrázek č. 12: Evidence učitelů (Zdroj: Vlastní zpracování)

Primárním klíčem tabulky učitel jsem zvolil ID učitele, které se bude generovat automaticky při vložení nového učitele do databáze, podobně jako v případě nového studenta. Tabulka předmětů obsahuje všechny vyučované předměty na škole a je propojena s tabulkou učitel pomocí pomocné tabulky vyučující. Každý učitel může vyučovat více předmětů a každý předmět může být vyučován více učiteli. Stejně jako funguje kontakt studenta a oprávnění u tabulky student, stejně tak funguje u tabulky učitel.

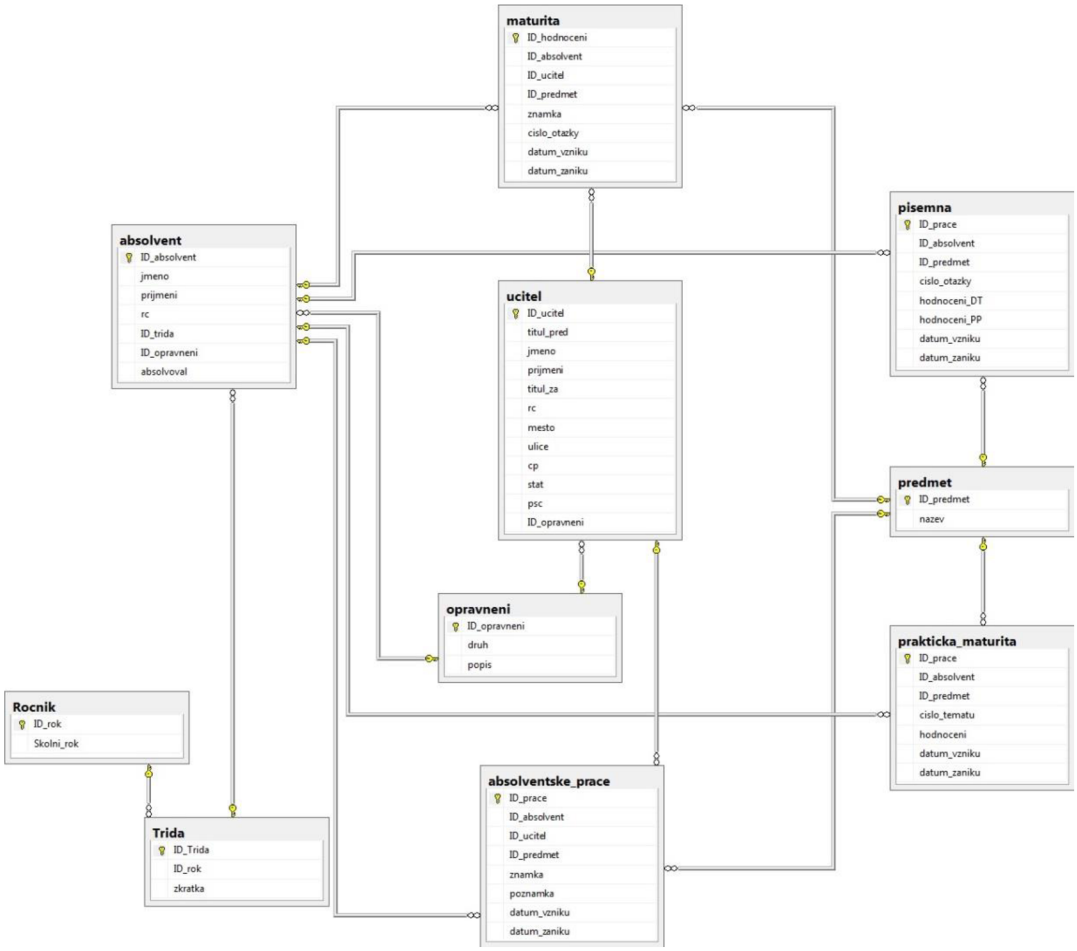
4.4.4 Evidence hodnocení studentů během jejich studia



Obrázek č. 13: Evidence hodnocení současných studentů (Zdroj: Vlastní zpracování)

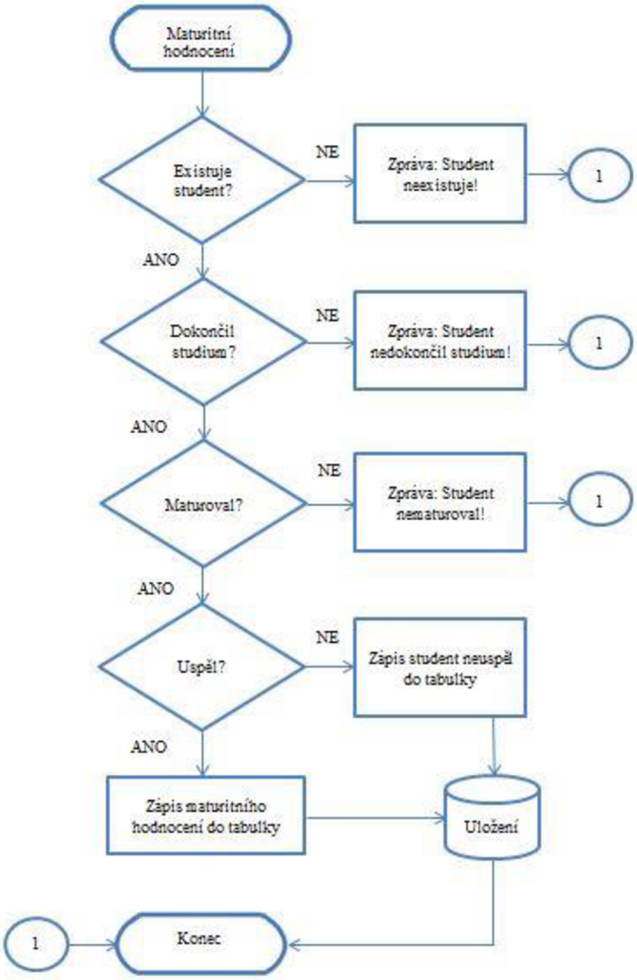
Primárním klíčem tabulky Žákovské práce jsem zvolil ID práce. Tabulka žákovské práce uchovává údaje o čtvrtletních a pololetních písemných pracích, které se musí uchovávat po dobu pěti let od vzniku práce. Poté, co student ukončí své studium, se data o žákovských pracích přesunou do tabulky absolventské práce a tam budou dále uložena.

4.4.5 Evidence absolventů a jejich hodnocení



Obrázek č. 14: Evidence hodnocení absolventů (Zdroj: Vlastní zpracování)

Kvůli uchovávání hodnocení absolventů vznikala tato databáze především. Hodnocení absolventů funguje tak, že po ukončení studia se student přesune z tabulky student i s vybranými údaji včetně jeho ID do tabulky absolvent. Primárním klíčem tabulky absolvent je ID absolventa, které se negeneruje automaticky, nýbrž ho získáme, jak jsem již zmínil, z tabulky student. Hodnocení je rozděleno do několika tabulek, jelikož se jedná o samostatné objekty. V tabulce maturita jsou uchovávány údaje o ústní maturitní zkoušce, jsou zde údaje o tom, kdo maturoval, z jakého předmětu, údaje o známce a hodnotiteli. Jsou zde také údaje o tom, jakou otázku si dotyčný vylosoval a tabulka obsahuje i datum, kdy byla maturita zapsána do databáze i datum, kdy bude z databáze vymazána. Tabulka písemná obsahuje údaje o písemných maturitních zkouškách. Tabulka praktická maturita, jak napovídá název tabulky, obsahuje údaje o praktické maturitní zkoušce. Tabulka absolventské práce plní funkci archivu pro žákovské práce. Pro lepší představu, jak funguje maturitní hodnocení, je na následující stránce zobrazen vývojový diagram maturitního hodnocení.



Obrázek č. 15: Vývojový diagram maturitního hodnocení (Zdroj: Vlastní zpracování)

4.5 Fyzický návrh

V této kapitole jsem vytvořil pomocí jazyka SQL samotnou databázi. Zdrojový kód můžete naleznout v příloze č. 1. Zde se budu věnovat pouze nejdůležitějším pohledům, procedurám a spouštím.

4.5.1 Pohledy

První pohled zobrazuje všechny studenty na škole, jejich ID, jméno a příjmení, třídu, ve které studují, školní rok a email. Druhý pohled zobrazuje všechny učitele, kteří ve škole vyučují. Je zde zobrazeno ID učitele, celé jméno, akademické tituly před a za jménem a telefonní číslo. Třetí pohled zobrazuje všechny absolventy školy. Opět se zobrazí ID absolventa, jméno a příjmení, třídu a datum ukončení studia. Tyto pohledy jsou velice jednoduché a slouží pro celkový přehled o všech osobách, které se ve škole pohybují. Složitější dotazy budu zpracovávat pomocí procedur a spouští.

```
create view studenti
as
select student.ID_student as 'ID studenta', student.prijmeni as 'Příjmení',
student.jmeno as 'Jméno', trida.zkratka as 'Třída', rocnik.Skolni_rok as 'Školní
rok', kontakt_studenta.email as 'Email'
from student
inner join Trida
on student.ID_Trida = Trida.ID_Trida
inner join Rocnik
on Rocnik.ID_rok = Trida.ID_rok
left join kontakt_studenta
on student.ID_student = kontakt_studenta.ID_student

select * from studenti

create view ucitele
as
select ucitel.ID_ucitel as 'ID_učitele',ucitel.titul_pred as 'Akademický titul',
ucitel.jmeno +' '+ ucitel.prijmeni as 'Jméno učitele', ucitel.titul_za as
'Vědecký titul', kontakt_ucitele.telefon as 'Telefon'
from ucitel
left join kontakt_ucitele
on ucitel.ID_ucitel = kontakt_ucitele.ID_ucitel

select * from ucitele

create view absolventi
as
select absolvent.jmeno +' '+ absolvent.prijmeni as 'Jméno absolventa',
absolvent.ID_absolvent as 'ID absolventa', trida.zkratka as 'Název třídy',
absolvent.absolvoval as 'Datum ukončení studia'
from absolvent, Trida
where absolvent.ID_trida = Trida.ID_Trida

select * from absolventi
```

4.5.2 Procedury

I když se studenti a učitelé budou do databáze převádět z IS Bakaláři, je dobré, aby měli učitelé možnost vložit nové údaje i manuálně. První procedura vkládá nového studenta do databáze. Procedura nejprve otestuje existenci studenta, jestliže student neexistuje, vloží ho do tabulky student a přiřadí do třídy, kde patří. Poté vloží údaje do tabulky kontakt_studenta, která je s tabulkou student propojená. Nakonec vypíše ID nového studenta. Jestliže student existuje, vypíše procedura hlášku, že student se již v databázi vyskytuje.

```
create procedure vloz_studenta
@jmeno varchar(20),
@prijmeni varchar(30),
@rodc varchar(10),
@mesto varchar(30),
@ulice varchar(25),
@cp varchar(10),
@stat varchar(20),
@psc char(5),
@trida_nazev varchar(6),
@skolni_rok varchar(9),
@opravneni int,
@telefon char(9),
@email varchar(40)
as
begin
declare @id_student int
declare @id_trida int
declare @id_rok int

 set @id_rok = (select rocnik.ID_rok from Rocnik where rocnik.Skolni_rok =
@skolni_rok)
 set @id_trida = (select trida.ID_Trida from Trida where trida.zkratka =
@trida_nazev and trida.ID_rok = @id_rok)
 set @id_student = (select top 1 id_student from student where student.jmeno =
@jmeno AND student.prijmeni = @prijmeni)

  if (@id_student is null)
   begin
    insert into
student(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_Trida,ID_opravneni)
    values
(@jmeno,@prijmeni,@rodc,@mesto,@ulice,@cp,@stat,@psc,@id_trida,@opravneni)

    SET @id_student = @@IDENTITY
    insert into kontakt_studenta(Telefon,email,ID_student)
    values (@telefon,@email,@id_student)
   end
  else
  print ('Student s ID: ' +convert(varchar(10),@id_student)+ ' se již v databázi
nachází. '  )
end
print ('ID nového studenta je: ' +convert(varchar(10),@id_student))

exec vloz_studenta 'Martin','Jurenka','9002105419','Frýdlant n. Ostravicí','U
řeky','10','Česká
republika','73805','4SA','2013/2014',3,'605789987','jurenka@yahoo.com'
```

Druhá procedura vkládá nového učitele a předmět, který vyučuje. Nejdříve otestuje existenci učitele, jestliže neexistuje, vloží nové údaje do tabulky učitel, poté vloží údaje do tabulky kontakt_učitele. Následně otestuje existenci předmětu, a když předmět neexistuje, zapíše ho mezi předměty a vypíše hlášku, že předmět byl přidán. Když již předmět v databázi existuje, vypíše procedura hlášení, že nebyl přidán, žádný předmět. Procedura dokáže přiřadit i již existující předmět k učiteli. Otestuje, jestli již takový předmět učitel nevyučuje a v případě, že ne, tak přidá záznam do tabulky vyučující a vypíše, že učitel byl přiřazen k předmětu. Jestliže už existuje jak učitel, tak předmět i vazba mezi nimi, vypíše procedura, že nedošlo k žádnému zápisu. Nakonec procedura vypíše ID učitele.

```
create procedure vloz_ucitele
@jmeno varchar(20),
@prijmeni varchar(30),
@rodc varchar(10),
@mesto varchar(30),
@ulice varchar(25),
@cp varchar(10),
@stat varchar(20),
@psc char(5),
@predmet_nazev varchar(20),
@opravneni int,
@telefon char(9),
@email varchar(40),
@titul_pred_jmenem varchar(10) = 'nevyplněno',
@titul_za_jmenem varchar(10) = 'nevyplněno'

as
declare @id_ucitel int
declare @id_predmet int

BEGIN
  set @id_predmet = (Select predmet.ID_predmet from predmet where predmet.nazev =
@predmet_nazev)
  set @id_ucitel = (select top 1 id_ucitel from ucitel where jmeno = @jmeno AND
prijmeni = @prijmeni)

   if (@id_ucitel IS NULL)
    begin
     insert into
ucitel(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_opravneni,titul_pred,titul_za
)
     values
(@jmeno,@prijmeni,@rodc,@mesto,@ulice,@cp,@stat,@psc,@opravneni,@titul_pred_jmene
m,@titul_za_jmenem)

     set @id_ucitel = @@IDENTITY

    end
   else
       begin
        print ('Tento učitel již existuje!')
     end
END
```

```
 if not exists (Select * from kontakt_ucitele where @id_ucitel =
kontakt_ucitele.ID_ucitel)
  begin
   insert into kontakt_ucitele(Telefon,email,ID_ucitel)
   values (@telefon,@email,@id_ucitel)
  end
 if not exists (Select predmet.nazev from predmet where @predmet_nazev =
predmet.nazev)
  begin

   insert into predmet(nazev)
   values (@predmet_nazev)

   set @id_predmet = (SELECT TOP 1 predmet.ID_predmet from predmet where
predmet.ID_predmet = @id_predmet)

   set @id_predmet = @@IDENTITY
   print ('Byl vložen nový předmět')

  end
 else

  print ('Nebyl vložen žádný nový předmět')

  if not exists (Select * from vyucujici where vyucujici.ID_ucitel = @id_ucitel
and vyucujici.ID_predmet = @id_predmet)
   begin
    print ('Učitel byl přiřazen k předmětu')

    insert into vyucujici(ID_ucitel,ID_predmet)
    values (@id_ucitel,@id_predmet)
   end
  else

   begin
    print ('Nebyl vložen žádný nový záznam')
   end

print ('ID nového učitele je: ' +convert(varchar(10),@id_ucitel))


exec vloz_ucitele
'Dagmar','Skotnicová','4511306887','Dobrá','Nádražní','59','Česká
republika','73810','Deskriptiva',2,'602568789','sko@sps.cz','Mgr.'
```

Procedura na vložení žákovských prací nejprve otestuje existenci studenta, poté existenci učitele a předmětu. Následně zjistí, jestli zadávaná známka je v rozsahu 1-5 a až potom vloží nový záznam.

```
create procedure vloz_zak_praci
@id_student int,
@id_ucitel int,
@predmet_nazev varchar(20),
@znamka int,
@poznamka varchar(500) = 'nevyplněno'
as
declare @id_predmet int
declare @znamkaa int
```

```
set @id_predmet = (Select predmet.ID_predmet from predmet where predmet.nazev =
@predmet_nazev)

if not exists (select student.ID_student from student where @id_student =
student.ID_student)
  begin
   print ('Student s ID: ' +convert(varchar(10),@id_student)+ ' neexistuje!')
  end
 else

  if not exists (Select ucitel.ID_ucitel from ucitel where @id_ucitel =
ucitel.ID_ucitel)
   begin
    print ('Učitel s ID: ' +convert(varchar(10),@id_ucitel)+ ' neexistuje!')
   end
  else

   if not exists (Select predmet.ID_predmet from predmet where @id_predmet =
predmet.ID_predmet)
    begin
        print ('Předmět : ' +@predmet_nazev+ ' neexistuje!')
       end
   else

   set @znamkaa = (select @znamka where @znamka > 0 and @znamka < 6)
     if (@znamkaa is null)
         begin
          print ('Hodnocení nebylo vloženo !')
         end
        else

insert into
zakovske_prace(ID_student,ID_ucitel,ID_predmet,znamka,poznamka,datum_vzniku)
values (@id_student,@id_ucitel,@id_predmet,@znamkaa,@poznamka,GETDATE())

exec vloz_zak_praci 5,3,'Matematika',2
```

Procedura na vytvoření absolventa otestuje existenci studenta, poté vloží vybrané údaje z tabulky student do tabulky absolvent. Následně přesune jeho žákovské práce mezi absolventské práce a nakonec smaže data z tabulky student, kontakt studenta, zástupce studenta a žákovské práce.

```
create procedure vytvor_absolventa
@id_student int
as
declare @jmeno varchar(20),
             @prijmeni varchar(30),
             @id_trida int,
             @id_opravneni int,
             @rc varchar(10)

             set @jmeno = (Select student.jmeno from student where
student.ID_student = @id_student)
             set @prijmeni = (Select student.prijmeni from student where
student.ID_student = @id_student)
             set @id_trida = (Select trida.ID_Trida from student, Trida where
student.ID_student = @id_student and student.ID_Trida = Trida.ID_Trida)
             set @id_opravneni = (select student.ID_opravneni from student where
student.ID_student = @id_student) + 2
```

```
            set @rc = (select student.rc from student where student.ID_student
= @id_student)

 if not exists (Select * from student where student.ID_student = @id_student)
  begin
   print ('Student s ID ' +convert(varchar(10),@id_student) +' neexistuje!')
  end
 else

insert into
absolvent(ID_absolvent,jmeno,prijmeni,rc,ID_trida,absolvoval,ID_opravneni)
values (@id_student,@jmeno,@prijmeni,@rc,@id_trida,GETDATE(),@id_opravneni)

insert into
absolventske_prace(ID_absolvent,ID_ucitel,ID_predmet,znamka,poznamka,datum_vzniku
)
select zakovske_prace.ID_student, zakovske_prace.ID_ucitel,
zakovske_prace.ID_predmet, zakovske_prace.znamka, zakovske_prace.poznamka,
zakovske_prace.datum_vzniku from zakovske_prace where zakovske_prace.ID_student =
@id_student
delete from kontakt_studenta where kontakt_studenta.ID_student = @id_student
delete from zakovske_prace where zakovske_prace.ID_student = @id_student
delete from zastupce_studenta where zastupce_studenta.ID_student = @id_student
delete from student where student.ID_student = @id_student


exec vytvor_absolventa 5
```

Procedura na vytvoření maturity otestuje existenci absolventa, učitele, předmětu a v případě potřeby vypíše příslušné hlášení. Nakonec otestuje, jestli je známka v rozsahu 1-5 a zapíše nové hodnocení ústní maturitní zkoušky.

```
create procedure vytvor_maturitu
@id_absolvent int,
@id_ucitel int,
@predmet_nazev varchar(20),
@cislo_otazky int,
@znamka int
as
declare @id_predmet int
declare @znamkaa int
 set @id_predmet = (select predmet.ID_predmet from predmet where @predmet_nazev =
predmet.nazev)

 if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_absolvent)
  begin
   print ('Student neexistuje')
  end
 else
  if not exists (Select * from ucitel where ucitel.ID_ucitel = @id_ucitel)
   begin
    print ('Učitel neexistuje')
   end
  else
   if not exists (Select * from predmet where predmet.ID_predmet = @id_predmet)
    begin
     print ('Předmět neexistuje')
    end
   else
    set @znamkaa = (select @znamka where @znamka > 0 and @znamka < 6)
```

```
     if (@znamkaa is null)
         begin
          print ('Známka musí být v rozsahu 1-5')
         end
     else

insert into
maturita(ID_absolvent,ID_ucitel,ID_predmet,cislo_otazky,znamka,datum_vzniku)
values (@id_absolvent,@id_ucitel,@id_predmet,@cislo_otazky,@znamka,GETDATE())

exec vytvor_maturitu 5,3,'Matematika',2,3
```

Procedura na vytvoření praktické maturity otestuje existenci absolventa, předmětu a rozsah známky a až poté vloží nové hodnocení.

```
create procedure vytvor_praktickou
@id_studenta int,
@predmet_nazev varchar(20),
@cislo_tematu int,
@hodnoceni int
as
declare @id_predmet int
declare @znamkaa int
 set @id_predmet = (select predmet.ID_predmet from predmet where @predmet_nazev =
predmet.nazev)

 if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_studenta)
  begin
   print ('Student neexistuje')
  end
 else
   if not exists (Select * from predmet where predmet.ID_predmet = @id_predmet)
    begin
     print ('Předmět neexistuje')
    end
   else
    set @znamkaa = (select @hodnoceni where @hodnoceni > 0 and @hodnoceni < 6)
     if (@znamkaa is null)
         begin
          print ('Známka musí být v rozsahu 1-5')
         end
     else

insert into
prakticka_maturita(ID_absolvent,ID_predmet,cislo_tematu,hodnoceni,datum_vzniku)
values (@id_studenta,@id_predmet,@cislo_tematu,@hodnoceni,GETDATE())

exec vytvor_praktickou 5,'Matematika',4,1
```

Procedura na vytvoření písemné maturity nejprve otestuje absolventa a předmět. Poté vloží nové údaje do tabulky písemná maturita. Proměnná @hodnocení_PP je nepovinná, protože ne z každého předmětu se píše písemná práce a proto je defaultní hodnota null.

```
create procedure vytvor_pisemnou
@id_absolventa int,
@predmet_nazev varchar(20),
@cislo_zadani int,
@hodnoceni_DT int,
```

```
@hodnoceni_PP int = null
as
declare @id_predmet int
 set @id_predmet = (select predmet.ID_predmet from predmet where @predmet_nazev =
predmet.nazev)

 if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_absolventa)
  begin
   print ('Absolvent neexistuje')
  end
 else
  if not exists (Select * from predmet where predmet.ID_predmet = @id_predmet)
   begin
    print ('Předmět neexistuje')
   end
  else

insert into
pisemna(ID_absolvent,ID_predmet,cislo_otazky,datum_vzniku,hodnoceni_DT,hodnoceni_
PP)
values
(@id_absolventa,@id_predmet,@cislo_zadani,GETDATE(),@hodnoceni_DT,@hodnoceni_PP)

exec vytvor_pisemnou 5,'Matematika',14,2
```

Procedura na výpis ústní maturitní zkoušky vypíše podle absolventova ID jeho jméno, příjmení, předmět, číslo otázky, hodnocení a kdo hodnotil.

```
create procedure vypis_mat
@id_absolventa int
as

if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_absolventa)
  begin
   print ('Absolvent neexistuje')
  end
 else

Select absolvent.jmeno as 'Jméno', absolvent.prijmeni as 'Příjmení',
predmet.nazev as 'Předmět', maturita.cislo_otazky as 'Otázka', maturita.znamka as
'Hodnocení', ucitel.prijmeni as 'Hodnotil'
from absolvent, predmet, maturita, ucitel
where absolvent.ID_absolvent = @id_absolventa and absolvent.ID_absolvent =
maturita.ID_absolvent and ucitel.ID_ucitel = maturita.ID_ucitel and
maturita.ID_predmet = predmet.ID_predmet

exec vypis_mat 5
```

Procedura na výpis písemné maturitní zkoušky vypíše podle absolventova ID jeho jméno, příjmení, předmět, číslo otázky, hodnocení didaktického testu a písemné práce.

```
create procedure vypis_pis
@id_absolventa int
as

if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_absolventa)
```

```
  begin
   print ('Absolvent neexistuje')
  end
 else

Select absolvent.jmeno as 'Jméno', absolvent.prijmeni as 'Příjmení',
predmet.nazev as 'Předmět', pisemna.cislo_otazky as 'Otázka',
pisemna.hodnoceni_DT as 'Hodnocení Didaktického testu', pisemna.hodnoceni_PP as
'Hodnocení Písemné práce'
from absolvent, predmet, pisemna
where absolvent.ID_absolvent = @id_absolventa and absolvent.ID_absolvent =
pisemna.ID_absolvent and pisemna.ID_predmet = predmet.ID_predmet

exec vypis_pis 5
```

Procedura na výpis praktické maturitní zkoušky otestuje existenci absolventa a poté vypíše jeho jméno, příjmení, předmět, číslo tématu a hodnocení.

```
create procedure vypis_prak
@id_absolventa int
as

if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_absolventa)
  begin
   print ('Absolvent neexistuje')
  end
 else

Select absolvent.jmeno as 'Jméno', absolvent.prijmeni as 'Příjmení',
predmet.nazev as 'Předmět', prakticka_maturita.cislo_tematu as 'Téma',
prakticka_maturita.hodnoceni as 'Hodnocení'
from absolvent, predmet, prakticka_maturita
where absolvent.ID_absolvent = @id_absolventa and absolvent.ID_absolvent =
prakticka_maturita.ID_absolvent and prakticka_maturita.ID_predmet =
predmet.ID_predmet

exec vypis_prak 5
```

Procedura na výpis absolventských prací. Opět otestuje existenci absolventa a poté vypíše požadované informace včetně data expirace.

```
create procedure vypis_abs
@id_absolventa int
as

if not exists (Select * from absolvent where absolvent.ID_absolvent =
@id_absolventa)
  begin
   print ('Absolvent neexistuje')
  end
 else

Select absolvent.jmeno as 'Jméno', absolvent.prijmeni as 'Příjmení',
predmet.nazev as 'Předmět', absolventske_prace.znamka as 'Hodnocení',
absolventske_prace.poznamka as 'Poznámky', ucitel.prijmeni as 'Hodnotil',
absolventske_prace.datum_zaniku as 'Datum expirace'
from absolvent, predmet, ucitel,absolventske_prace
```

```
where absolvent.ID_absolvent = @id_absolventa and absolvent.ID_absolvent =
absolventske_prace.ID_absolvent and absolventske_prace.ID_predmet =
predmet.ID_predmet and absolventske_prace.ID_ucitel = ucitel.ID_ucitel

exec vypis_abs 5
```

4.5.3 Spouště

Tyto jednoduché spouště mají za úkol při vložení nového řádku připočítat k aktuálnímu datu počet let, kdy vyprší povinná minimální doba archivace těchto dat.

```
create trigger expirace_mat
on maturita
for insert
as
update maturita
set datum_zaniku = DATEADD(year,45,datum_vzniku)


create trigger expirace_abs
on absolventske_prace
for insert
as
update absolventske_prace
set datum_zaniku = DATEADD(year,5,datum_vzniku)

create trigger expirace_prak
on prakticka_maturita
for insert
as
update prakticka_maturita
set datum_zaniku = DATEADD(year,45,datum_vzniku)

create trigger expirace_pis
on pisemna
for insert
as
update pisemna
set datum_zaniku = DATEADD(year,45,datum_vzniku)
```

5. Závěr

Cílem mé práce bylo navrhnout databázi, která bude archivovat hodnocení studentů za dobu jejich studia na Střední průmyslové škole ve Frýdku-Místku. Důraz byl kladen na uchovávání údajů, které se používají pro výpis maturitního protokolu a uchovávání údajů o žákovských pracích, které se archivují 5 let a v případě maturitního protokolu 45 let. Tato databáze vychází z analýzy současného stavu školy. Práce byla konzultována se správcem archivu, tak aby vyhovovala veškerým požadavkům.

V analýze současného stavu jsem popsal, jak funguje školní archiv v současné době. Na základě této analýzy jsem vypracoval návrh databáze, která může nahradit papírovou formu vedení archivu. Poté jsem tento návrh implementoval pomocí jazyka SQL a přidal různé pohledy, procedury a spouště, tak aby se s databází, co nejlépe pracovalo. Tím jsem splnil všechny vytyčené cíle mé bakalářské práce.

Seznam použitých zdrojů

[1] KOCH, M. *Datové a funkční modelování.* Brno: CERM, 2004.
ISBN 80-214-2724-8.

[2] OPPEL, A. *SQL bez předchozích znalostí.* Brno: Computer Press, 2012.
ISBN 978-80-251-1707-1.

[3] LACKO, L. *1001 tipů a triků pro SQL.* Brno: Computer Press, 2011.
ISBN 978-80-251-3010-0.

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

[5] CONOLLY, T., C. BEGG a R. HOLOWCZAK. *Mistrovství – Databáze: Profesionální průvodce tvorbou efektivních databází.* 1. vyd. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.

[6] LUHAN, J. Databázové systémy a metodologie návrhu. Vut.cz [online]. Vysoké učení technické v Brně, Fakulta podnikatelská: 2011 [cit. 2014-01-30]. Dostupné z: http://luhan.comlu.com/DBS/doc/P/02/02.pdf

[7] SPSOAFM. Historie. Spsoafm.cz [online]. [cit. 2014-01-31]. Dostupné z: http://www.oafm.cz/web/stredni-prumyslova-skola-2/historie-sps2

[8] ARCHIV HLAVNÍHO MĚSTA PRAHY (AHMP). Vzorový spisový a skartační řád pro základní a střední školy. Ahmp.cz [online]. 2004 [cit. 2014-05-30]. Dostupné z: www.ahmp.cz/page/docs/skoly.pdf

[9] NÚV. Rvp. Rvp.cz [online]. [cit. 2014-05-06]. Dostupné z: http://clanky.rvp.cz/clanek/c/Z/8019/skolni-informacni-systemy.html

[10] MOODLE. Moodle. Moodle.org [online]. [cit. 2014-05-06]. Dostupné z: http://docs.moodle.org/archive/cs/Co_je_Moodle

[11] BAKALÁŘI. Bakalari. Bakalari.cz [online]. [cit. 2014-05-06]. Dostupné z: http://www.bakalari.cz/programy.aspx

[12] MOLINARO, A. *SQL kuchařka programátora.* Brno: Computer press, 2009. ISBN 978-80-251-2617-2.

[13] LUHAN, J. Metodologie návrhu databáze II. Vut.cz [online]. Vysoké učení technické v Brně, Fakulta podnikatelská: 2011 [cit. 2014-05-07]. Dostupné z: http://luhan.comlu.com/DBS/doc/P/04/04.pdf

[14] BEŇOVÁ, J. Ústní sdělení. Střední průmyslová škola, Obchodní akademie a Jazyková škola s právem státní jazykové zkoušky, Frýdek-Místek, příspěvková organizace, 28. října 1598, 73801 Frýdek-Místek. 9.5.2014.

Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha č. 1 Zdrojový kód SQL

```
use Bakalarska_prace
go
-- Tabulka oprávnění
Create table opravneni
(
ID_opravneni int identity (1,1) primary key not null,
druh varchar(20) not null,
popis varchar(100)
)
go
-- Tabulka Ročník
Create table Rocnik
(
ID_rok int identity (1,1) primary key not null,
Skolni_rok varchar(9) not null
)
go
-- Tabulka Tříd
Create table Trida
(
ID_Trida int identity (1,1) primary key not null,
ID_rok int not null,
zkratka varchar(6) not null,
Constraint FK_trida_rocnik foreign key (ID_rok) references Rocnik(ID_rok)
)
go
-- Tabulka studentů
create table student
(
ID_student int identity (1,1) primary key not null,
jmeno varchar(20) not null,
prijmeni varchar(30) not null,
rc char(10) not null,
mesto varchar(30) not null,
ulice varchar(25) not null,
cp varchar(10) not null,
stat varchar(20) not null,
psc char(5) not null,
ID_Trida int not null,
ID_opravneni int not null,
Constraint FK_student_trida foreign key(ID_trida) references Trida(ID_Trida),
Constraint FK_student_opravneni foreign key (ID_opravneni) references
Opravneni(ID_opravneni)
)
go
-- Tabulka učitelů
create table ucitel
(
ID_ucitel int identity (1,1) primary key not null,
titul_pred varchar(10) default 'neuvedeno',
jmeno varchar(20) not null,
prijmeni varchar(30) not null,
titul_za varchar(10) default 'neuvedeno',
rc char(10) not null,
mesto varchar(30) not null,
ulice varchar(25) not null,
cp varchar(10) not null,
stat varchar(20) not null,
psc char(5) not null,
ID_opravneni int,
```

```
Constraint FK_ucitel_opravneni foreign key (ID_opravneni) references
opravneni(ID_opravneni)
)
go
-- Tabulka předmětů
create table predmet
(
ID_predmet int identity (1,1) primary key not null,
nazev varchar(20) not null
)
go
-- Tabulka vyučujících
create table vyucujici
(
ID_ucitel int not null,
ID_predmet int not null,
Constraint PK_vyucujici primary key(ID_ucitel, ID_predmet),
Constraint FK_vyucujici_ucitel foreign key (ID_ucitel) references
ucitel(ID_ucitel),
Constraint FK_vyucujici_predmet foreign key (ID_predmet) references
predmet(ID_predmet)
)
go
-- Tabulka zákonných zástupců studentů
create table zastupce
(
ID_zastupce int identity (1,1) primary key not null,
titul_pred varchar(10) default 'neuvedeno',
jmeno varchar(20) not null,
prijmeni varchar(30) not null,
titul_za varchar(10) default 'neuvedeno',
rc char(10) not null,
mesto varchar(30) not null,
ulice varchar(25) not null,
cp varchar(10) not null,
stat varchar(20) not null,
psc char(5) not null,
ID_opravneni int not null,
Constraint FK_zastupce_opravneni foreign key (ID_opravneni) references
opravneni(ID_opravneni)
)
go
-- Tabulka přiřazující zástupce ke studentům
Create table zastupce_studenta
(
ID_zastupce int not null,
ID_student int not null,
Constraint PK_zastupce_studenta primary key (ID_zastupce, ID_student),
Constraint FK_zastupce_studenta_zastupce foreign key (ID_zastupce) references
zastupce(ID_zastupce),
Constraint FK_zastupce_studenta_student foreign key (ID_student) references
student(ID_student)
)
go
-- Tabulka kontaktů na studenta
Create table kontakt_studenta
(
Telefon char(9) primary key not null,
email varchar(40),
ID_student int not null,
Constraint FK_kontakt_studenta_student foreign key (ID_student) references
student(ID_student)
)
```

```
go
-- Tabulka kontaktů na učitele
create table kontakt_ucitele
(
telefon char(9) primary key not null,
email varchar(40),
ID_ucitel int not null,
Constraint FK_kontakt_ucitele_ucitel foreign key (ID_ucitel) references
ucitel(ID_ucitel)
)
go
-- Tabulka absolventů
create table absolvent
(
ID_absolvent int primary key not null,
jmeno varchar(20) not null,
prijmeni varchar(30) not null,
rc char(10) not null,
ID_trida int not null,
ID_opravneni int not null,
absolvoval date not null,
Constraint FK_absolvent_trida foreign key (ID_trida) references trida(ID_trida),
Constraint FKabsolvent_opravneni foreign key (ID_opravneni) references
opravneni(ID_opravneni)
)
go
-- Tabulka ústních maturitních hodnocení absolventů
create table maturita
(
ID_hodnoceni int identity (1,1) primary key not null,
ID_absolvent int not null,
ID_ucitel int not null,
ID_predmet int not null,
znamka int not null,
cislo_otazky int not null,
datum_vzniku date not null,
datum_zaniku date,
Constraint FK_maturita_absolvent foreign key (ID_absolvent) references
absolvent(ID_absolvent),
Constraint FK_maturita_ucitel foreign key (ID_ucitel) references
ucitel(ID_ucitel),
Constraint FK_maturita_predmet foreign key (ID_predmet) references
predmet(ID_predmet)
)
go
-- Tabulka čtvrtletních a pololetních písemných prací studentů během studia
create table zakovske_prace
(
ID_prace int identity (1,1) primary key not null,
ID_student int not null,
ID_ucitel int not null,
ID_predmet int not null,
znamka int not null,
poznamka varchar(500),
datum_vzniku date not null,
Constraint FK_zakovske_prace_student foreign key (ID_student) references
student(ID_student),
Constraint FK_zakovske_prace_ucitel foreign key (ID_ucitel) references
ucitel(ID_ucitel),
Constraint FK_zakovske_prace_predmet foreign key (ID_predmet) references
predmet(ID_predmet)
)
go
```

```
-- Tabulka čtvrtletních a pololetních písemných prací absolventů
create table absolventske_prace
(
ID_prace int identity (1,1) primary key not null,
ID_absolvent int not null,
ID_ucitel int not null,
ID_predmet int not null,
znamka int not null,
poznamka varchar(500),
datum_vzniku date not null,
datum_zaniku date,
Constraint FK_absolventske_prace_absolvent foreign key (ID_absolvent) references
absolvent(ID_absolvent),
Constraint FK_absolventske_prace_ucitel foreign key (ID_ucitel) references
ucitel(ID_ucitel),
Constraint FK_absolventske_prace_predmet foreign key (ID_predmet) references
predmet(ID_predmet)
)
go
-- Tabulka hodnocení písemných maturitních prací
create table pisemna
(
ID_prace int identity (1,1) primary key not null,
ID_absolvent int not null,
ID_predmet int not null,
cislo_otazky int not null,
hodnoceni_DT int,
hodnoceni_PP int,
datum_vzniku date not null,
datum_zaniku date,
Constraint FK_pisemna_absolvent foreign key (ID_absolvent) references
absolvent(ID_absolvent),
Constraint FK_pisemna_predmet foreign key (ID_predmet) references
predmet(ID_predmet)
)
go
-- Tabulka hodnocení praktické maturitní zkoušky
Create table prakticka_maturita
(
ID_prace int identity (1,1) primary key not null,
ID_absolvent int not null,
ID_predmet int not null,
cislo_tematu int not null,
hodnoceni int not null,
datum_vzniku date not null,
datum_zaniku date,
Constraint FK_prakticka_maturita_absolvent foreign key (ID_absolvent) references
absolvent(ID_absolvent),
Constraint FK_prakticka_maturita_predmet foreign key (ID_predmet) references
predmet(ID_predmet)
)

-- Vložení dat
insert into opravneni(druh,popis)
Values ('Administrátor','Hlavní správce databáze s veškerými právy')
insert into opravneni(druh,popis)
Values ('Učitel','Učitel s právem čtení a zápisu')
insert into opravneni(druh,popis)
Values ('Student','Student s právem čtení')
insert into opravneni(druh,popis)
Values ('Zástupce','Zástupce s právem čtení dat svého studenta')
insert into opravneni(druh,popis)
Values ('Absolvent','Absolvent s omezeným právem čtení')
```

```
go
insert into Rocnik(Skolni_rok)
Values ('2012/2013')
insert into Rocnik(Skolni_rok)
Values ('2013/2014')
go
Insert into Trida(ID_rok,zkratka)
Values (1,'4SA')
Insert into Trida(ID_rok,zkratka)
Values (1,'4SB')
Insert into Trida(ID_rok,zkratka)
Values (2,'4SA')
Insert into Trida(ID_rok,zkratka)
Values (2,'4SC')
go
Insert into student
(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_Trida,ID_opravneni)
Values ('František','Novák','9102035629','Frýdek-Místek','Anenská','4268','Česká
republika','73801',1,3)
Insert into student
(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_Trida,ID_opravneni)
Values ('Josef','Chládek','9205065049','Frýdek-Místek','Bruzovská','126','Česká
republika','73801',2,3)
Insert into student
(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_Trida,ID_opravneni)
Values ('Tomáš','Buben','9004105409','Třinec','Frýdecká','2526','Česká
republika','73820',3,3)
Insert into student
(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_Trida,ID_opravneni)
Values ('Kamil','Buben','9210105405','Třinec','Frýdecká','2526','Česká
republika','73820',3,3)
go
Insert into
ucitel(titul_pred,jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_opravneni)
Values ('Mgr.','Tomáš','Kvapil','6854231458','Frýdek-Místek','Kapitána
Nálepky','13','Česká republika','73801',2)
Insert into ucitel
(titul_pred,jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_opravneni)
Values ('Ing.','Karel','Přísný','7554851458','Ostrava','Místecká','1315','Česká
republika','73820',2)
Insert into ucitel
(titul_pred,jmeno,prijmeni,titul_za,rc,mesto,ulice,cp,stat,psc,ID_opravneni)
Values
('Ing.','Jan','Klavír','Phd','6048131458','Havířov','Ostravská','153','Česká
republika','73801',2)
go
Insert into predmet(nazev)
Values ('Matematika')
Insert into predmet(nazev)
Values ('Chemie')
Insert into predmet(nazev)
Values ('Fyzika')
Insert into predmet(nazev)
Values ('Angličtina')
Insert into predmet(nazev)
Values ('Stroje')
go
Insert into vyucujici(ID_predmet,ID_ucitel)
Values (1,1)
Insert into vyucujici(ID_predmet,ID_ucitel)
Values (2,2)
Insert into vyucujici(ID_predmet,ID_ucitel)
Values (3,3)
```

```
go
insert into
zastupce(titul_pred,jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_opravneni)
values ('Ing.','Jindřich','Novák','6854231458','Frýdek-
Místek','Anenská','4268','Česká republika','73801',4)
insert into zastupce(jmeno,prijmeni,rc,mesto,ulice,cp,stat,psc,ID_opravneni)
values ('Gábina','Nováková','6754231458','Frýdek-Místek','Anenská','4268','Česká
republika','73801',4)
go
insert into zastupce_studenta(ID_student,ID_zastupce)
values (1,1)
insert into zastupce_studenta(ID_student,ID_zastupce)
values (1,2)
go
insert into kontakt_studenta(Telefon,email,ID_student)
values ('604568407','novak@seznam.cz',1)
insert into kontakt_studenta(Telefon,email,ID_student)
values ('123456789','chladek@email.cz',2)
insert into kontakt_studenta(Telefon,email,ID_student)
values ('987654321','buben123@seznam.cz',3)
go
insert into kontakt_ucitele(Telefon,email,ID_ucitel)
values ('123879568','kvapil@yahoo.com',1)
insert into kontakt_ucitele(Telefon,email,ID_ucitel)
values ('546987458','prisny@quick.cz',2)
insert into kontakt_ucitele(Telefon,email,ID_ucitel)
values ('546231456','klavir@yseznam.cz',3)
go
insert into zakovske_prace(ID_student, ID_ucitel, ID_predmet, znamka, poznamka,
datum_vzniku)
values (1,1,3,2,'Práce obsahovala menší chyby, které neměly vliv na celkový
výsledek.', GETDATE())
insert into zakovske_prace(ID_student, ID_ucitel, ID_predmet, znamka, poznamka,
datum_vzniku)
values (2,2,4,1,'Perfektní práce bez chyb!', GETDATE())
go
```