VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH DATABÁZE PRO PODPORU PERSONÁLNÍ AGENDY

DESIGN OF DATABASE FOR SUPPORT OF PERSONAL AGENDA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ SKOUPIL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN LUHAN, Ph.D.
SUPERVISOR

BRNO 2014

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2014/2015
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Skoupil 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 databáze pro podporu personální agendy

v anglickém jazyce:

Design of Database for Support of Personal Agenda

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:

BEGG, C., R. HOLOWCZAK a T. CONOLLY. Mistrovství - Databáze: Profesionální průvodce tvorbou efektivních databází. Praha: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.
HOTEK, M. Microsoft SQL server 2008: Krok za krokem. 1. vyd. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6.
CHURCHER, C. Beginning Database Design. 2nd ed. New York City: Apress, 2012. 252 p. ISBN 978-1-4302-4209-3.
KOCH, M. Datové a funkční modelování. 1. vyd. Brno: CERM, 2004. 108 s. ISBN 80-214-2724-8.
KŘÍŽ, J. a P. DOSTÁL. Databázové systémy. 1. vyd. Brno: CERM, 2005. 111 s. ISBN 80-214-3064-8.
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. Jan Luhan, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/2015.

L.S.

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

V Brně, dne 1.12.2014

Abstrakt

Tato bakalářská práce pojednává o návrhu databázového systému pro personální agendu. Má za úkol zjednodušit a zefektivnit dosavadní práci akademických pracovníků v oblasti řízení lidských zdrojů. V první části jsou rozvedeny teoretické poznatky nutné pro shrnutí dané problematiky. Tyto poznatky jsou dále použity při popisu a analýze současné situace. Na těchto základech poté stojí celý vlastní návrh. Návrh je nakonec otestován, čímž demonstruje svou funkčnost.

Abstract

This bachelor thesis concern about proposal of database system for personal administration. The main task here is make actual work of academic workers more effective and easier to work with human resources management. The first part presents the theoretical foundation required to make this issues easier to understand. This knowledge is used in subscribe and analysis of actual situation and whole proposal is based on that. Proposal's functionality is demonstrate by tests at the end.

Klíčová slova

SQL, databáze, relace, integrita, normalizace, MS SQL server, datové modelování

Key words

SQL, database, integrity, normalization, MS SQL server, data simulation

Bibliografická citace práce

SKOUPIL, O. *Návrh databáze SQL pro personální agendu*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 67 s. Vedoucí bakalářské práce Ing. Jan Luhan, 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 ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 29. prosince 2014

……

Poděkování

Chtěl bych poděkovat vedoucímu práce, panu Ing. Janu Luhanovi, Ph.D., za čas a cenné rady poskytnuté při odborném vedení práce.

Obsah

Úvod

Tato bakalářská práce je zaměřena na návrh databázového systému pro evidenci personální agendy akademických pracovníků. Její struktura je rozdělena na několik částí.

V teoretické části se zaměřuji na potřebnou znalost jazyka SQL a přehled potřebných informací důležitých pro pochopení dané problematiky. Tyto znalosti jsou dále nutné pro vznik adekvátního systému vzhledem k řešenému problému.

Navazující kapitola je zaměřena na sjednocení požadavků na systém. Požadavky jsou zde specifikovány jak z globálního hlediska, tak i z pohledu dílčích částí nutných pro vznik systému.

Tyto požadavky jsou dále podrobeny podrobnému rozboru v následující kapitole.

Poslední, největší a nejdůležitější část se zabývá vlastním návrhem samotného systému. Postupuje od návrhu schématu a jeho dílčích částí. Následně se zaobírá návrhem jednotlivých entit a relací v rámci jednotlivých agend.

Systém jako takový nemůže spolehlivě fungovat bez definování automatizovaných procesů, proto jsou dále popsány funkce, procedury a triggery zajišťující plynulý a bezpečný chod databáze.

Pro uživatele je přímá práce s databází neefektivní, jelikož se v ní špatně orientuje a nezná strukturu systému. Z tohoto důvodu jsou vytvořeny pohledy, jež umožňují přirozenější získávání informací. Tyto pohledy jsou uzpůsobeny jak pro jednotlivé uživatele, tak i pro správce lidských zdrojů.

Z důvodu zahájení testování je nutné naplnit tabulky orientačními daty pro předejití řady vyskytujících se chyb. Zároveň je nutné zajistit data, jež budou dále potřebná pro reálný chod databáze, a to především v jednotlivých číselnících.

Podstatnou částí systému je také zabezpečení databáze nejen před možnými úniky a poškozením dat. Toho má být docíleno za pomoci předdefinovaných rolí, které definují prostor, v němž se uživatel dané skupiny může pohybovat. Dále zálohováním dat

umožníme rychlou obnovu ztracené či poškozené části z daného časového úseku. Navíc se očekává i archivace dat, jejímž hlavním účelem je dlouhodobé uchování dat, které nejsou potřeba pro každodenní využití.

V závěru práce se zaobírám rozmanitými přínosy návrhu řešení, ale i nevýhodami systému a možnostmi budoucího využití.

1 Cíl a metodika práce

1.1 Vymezení problému

Z důvodu zefektivnění plánování výuky je důležité vhodně využít lidské zdroje, a to jak z hlediska personalistiky, tak i pedagogické způsobilosti v rámci akademických institucí.

Na druhou stranu se nesmí zapomínat na odbornost jednotlivých pedagogů a je vhodné využít jejich profesní způsobilosti. Každý pedagog má odlišná zaměření. Je proto podstatné zaznamenávat data o způsobilosti daného jedince v jednotlivých oborech a usměrnit je v přehledné informace.

Na základě tohoto problému je vhodné vytvořit systém, jenž má sloužit především managementu pro ucelený přehled o lidských zdrojích, které jsou jim k dispozici. Otevřou se tak nové možnosti pro správu lidských zdrojů, ale i plánování výuky. Přičemž výstupem tohoto systému se předpokládají přehledné souhrny dat jednotlivých částí se zaměřením na tuto problematiku. Použitelnost databáze lze kupříkladu demonstrovat i automatickou tvorbou strukturovaných životopisů jednotlivých zaměstnanců.

1.2 Cíle práce

Hlavním cílem této bakalářské práce je vytvořit podpůrnou databázi akademických pracovníků a doktorandů. Ta se bude skládat z jednotlivých agend a podpůrných struktur, které navazují na danou problematiku. Databáze musí splňovat formální kritéria, jako jsou například normalizační formy. Projekt by měl být následně uveden do testovacího provozu pro korekci možných chyb.

2 Teorie

Obsahuje souhrnné informace z oblasti teorie databází a jazyka SQL. Tyto poznatky jsou dále použity nejenom při vlastním návrhu databázové struktury.

Nejprve se zaměřím na základní pojmy, jež jsou nutné pro pochopení souborů databázových znalostí. Následně přiblížím, co je SQL Server a samotný jazyk SQL, jenž představuje významnou součást praktické části. Teoretická část je ukončena popisem problematiky zabezpečení chodu a ukládání databáze.

2.1 Základní pojmy a definice

2.1.1 Data

Pod pojmem data si lze představit surová fakta, jež jsou svým způsobem důležitá pro jednotlivce či organizaci. Data, která prošla zpracováním nebo dostala určitou strukturu, jsou nazývána informacemi. [2]

2.1.2 Informace

Informace jsou tedy data zpracovaná do podoby, v níž mají určitý význam či smysl. Na informaci lze nahlížet z různých hledisek. Informaci lze chápat jako zprávu, která splňuje tři požadavky. Prvním je syntaktická relevance. Subjekt, který zprávu přijímá, musí být schopen ji detekovat a rozumět jí. Druhým požadavkem je sémantická relevance. Subjekt musí vědět, co zpráva znamená, co vypovídá o něm a jeho okolí. Třetím požadavkem je pragmatická relevance. Zpráva musí mít pro přijímající subjekty nějaký význam. [8]

2.2 Databáze

Databáze je kolekce vzájemně souvisejících datových položek, které jsou spravovány jako jediná jednotka. Tato definice je úmyslně široká, protože mezi různými výrobci softwaru, kteří poskytují databázové systémy, existuje mnoho variací. Společnost Oracle Corporation například definuje svou databázi jako kolekci fyzických souborů, které jsou řízeny jedinou instancí databázového softwaru, zatímco Microsoft definuje databázi SQL Server jako kolekci tabulek s daty a jinými objekty. [14]

2.2.1 Pohledy

Vytváří se pro zvýšení pohodlí koncového uživatele. Každý pohled zobrazuje uživateli vybranou množinu dat, neboť kompletní systém je pro něj příliš komplikovaný a zahrnuje data, jež vůbec nepotřebuje. Samotný pohled je virtuální tabulka, která v databázi nemusí vůbec existovat. Je generován z databázových tabulek, na kterých je založen, kdykoli se k němu přistupuje. Mechanismus pohledů poskytuje každému uživateli upravený pohled na databázi. [2]

Pohledy zprostředkovávají pohled na vybrané atributy i z více tabulek současně. Vyfiltrování nepotřebných údajů vede k smysluplnějšímu přístupu a je tedy pro tuto práci velmi důležité. Předpokládá se, že výstupy v podobě pohledů budou použity v aplikační části systému. Za pomoci aplikační části tak lze dosáhnout daleko lepšího interaktivního a intuitivního ovládání databázového systému. Uživatel tak nemusí vykonávat žádné příkazy, ani si je nemusí pamatovat. Skrze aplikační formu mu budou dané pohledy jednoduše poskytnuty.

2.2.2 Schémata a instance

Jedná se o celkový popis databáze, který je specifikován během návrhu databáze. Data v databázi v kterémkoli konkrétním časovém okamžiku se nazývají instance databáze. Přestože se data v průběhu času neustále mění, schéma zůstává takřka neměnné. [2]

Schéma databáze nám ukazuje, jak jsou jednotlivé entity uspořádány a touto formou jsou po částech prezentovány ve vlastním návrhu řešení.

2.3 Relační datový model

K nejpoužívanějším datovým modelům v současnosti patří relační model. Vzniká několik entit spojených dohromady pomocí relačního klíče. Toto spojení není trvalé, ale vzniká v okamžiku, kdy potřebujeme mít společně k dispozici data ze všech spojených tabulek, přičemž zaniká, když práci s modelem ukončíme. [8]

2.3.1 Relace

Pojem relace je vyjádřen z matematické terminologie spolu s teorií množin a predikátové logiky. Relace je množina smysluplných spojení mezi zúčastněnými entitami. Stejně jako u entit by mělo být každé spojení jedinečně identifikovatelné v rámci této množiny. Jedinečně identifikovatelné spojení se nazývá výskyt relace. Každá relace je označena názvem, který popisuje její funkci. Například entita Člověk je spojena s entitou Práce, prostřednictvím relace Pracuje. [2]

2.3.2 Klíče relace

Jelikož je nevhodné uchovávat více stejných záznamů v databázi, je nutné zajistit, aby každý záznam byl jedinečný. Toho lze docílit, pokud se každému záznamu přiřadí určitý klíč.

2.3.2.1 Kandidátní klíč

Skládá se z jednoho nebo více atributů, které jsou jednoznačné pro daný záznam v tabulce. Kandidátní klíč musí být jednoznačný a neredukovatelný. Volí se na základě dostupných atributů, případně je možno vytvořit i klíč umělý. Kandidátních klíčů může být vždy několik. Funkcí kandidátního klíče je tedy vyjádřit ostatní pole v tabulce. [7]

2.3.2.2 Primární klíč

Zvolený klíč z množiny kandidátních klíčů dané tabulky se nazývá primární klíč. Jako takový musí splňovat podmínky kandidátního klíče.

2.3.2.3 Cizí klíč

Cizí klíč se skládá z jednoho nebo více atributů. Tyto atributy jsou shodné s kandidátním, většinou však přímo primárním klíčem z jiné nebo stejné tabulky. [2]

2.3.3 Relační datová struktura

Relační model dat se skládá z datové struktury. Ta obsahuje pět hlavních složek: relaci, atribut, datovou n-tici, doménu a relační databázi. Relaci reprezentujeme jako tabulku, v níž řádky tabulky odpovídají jednotlivým datovým n-ticím a sloupce tabulky odpovídají atributům. Atributy prezentují sloupce tabulek a nabývají určitých datových typů. Všechny záznamy v tomto sloupci nabývají stejný datový typ. Pod relační databází si proto představíme kolekci normalizovaných tabulek. [2]

2.3.4 Relační integrita

Protože každý sloupec je spojen s doménou, existují omezení (nazývaná doménová omezení) množiny hodnot přípustných pro daný sloupec tabulky. Navíc existují dvě důležitá integritní pravidla neboli omezení, která se vztahují na všechny instance databáze.

Tato dvě základní pravidla pro relační databázový model jsou známa jako entitní integrita a referenční integrita. [2]

2.3.5 Entitní integrita

Označuje, že hodnota primárního klíče nesmí být neznámá nebo neplatná. [2]

2.3.6 Referenční integrita

Pokud existuje v tabulce cizí klíč, musí buď hodnota cizího klíče odpovídat hodnotě některého záznamu v domovské tabulce, nebo musí mít cizí klíč prázdnou hodnotu. [2]

2.3.7 Entita

Entitou je nazývána množina objektů z reálného života, které jsou zachyceny v datovém modelu. Každý objekt, který je možné jedinečně identifikovat v rámci této množiny, se nazývá výskyt entity. Entita existuje nezávisle a může představovat objekt s fyzickou („skutečnou") existencí nebo objekt existující pojmově („abstraktně"). [2]

2.4 Normalizace dat

Normalizace dat je technika, jež obsahuje řadu pravidel, jejímž cílem je omezení nadbytečných hodnot databáze. [2]

2.4.1 První normální forma

Tabulka je v první normální formě, pokud je každý její atribut dále nedělitelný. Výjimkou je, pokud se nepotřebují jednotlivé části tohoto atributu (například v některých případech adresa). [4]

2.4.2 Druhá normální forma

Tabulka, která je v první normální formě a ve které jsou hodnoty každého sloupce, který není součástí primárního klíče, determinovány všemi hodnotami sloupců, které tvoří primární klíč.

2.4.3 Třetí normální forma – tranzitivní závislost

Relace je ve třetí normální formě, jestliže již je v první a druhé normální formě a každý neklíčový atribut jejího schématu není závislý na žádném klíči. [7]

2.4.4 Boyce–Coddova normální forma

Boyce–Coddova normální forma se pokládá za variaci třetí normální formy. Pro její platnost nesmí žádný neklíčový atribut určovat hodnotu žádného jiného atributu. Tabulka je tedy v Boyce-Coddově normální formě, pokud by každý atribut mohl být primárním klíčem. [7]

Zároveň zde musí platit několik podmínek:

- Relace musí mít více kandidátních klíčů.
- Minimálně 2 kandidátní klíče musí být složené z více atributů.
- Některé složené kandidátní klíče musí mít společný atribut.

2.4.5 Čtvrtá normální forma

Relace je ve čtvrté normální formě, pokud je v Boyce–Coddově normální formě a navíc všechny vícehodnotové závislosti jsou zároveň funkčními závislostmi z kandidátních klíčů. [8]

2.4.6 Pátá normální forma

Týká se případu spojené závislosti, která vyjadřuje cyklické omezení. Například pokud je relace jedna spojena s relací číslo dvě, relace dva je spojena s relací číslo tři a relace tři je opět spojena zpětně s relací číslo jedna. V tomto případě všechny tři entity musí být součástí stejného vektoru hodnot. [8]

2.5 SQL

SQL, zkráceně Structured Query Language (Strukturovaný dotazovací jazyk), slouží nejen pro tvorbu dotazů v databázích. Dále také umožňuje zakládání tabulek, ošetření vstupů, sdílení dat nebo zabezpečení databáze.

O SQL by se mělo správně mluvit jako o podjazyku, protože neobsahuje žádné prostředky pro manipulaci s obrazovkami a pro uživatelský vstup a výstup. Jeho hlavním účelem je poskytovat standardní metodu přístupu k databázi, nezávisle na jazyku, v němž je napsána zbývající část databázové aplikace. [9]

2.5.1 Datové typy SQL

V databázi je každý atribut (sloupec tabulek) určitého datového typu. Datový typ je určen podle druhu hodnot a největší velikosti/rozpětí hodnot, jež budou v atributu obsaženy. Například při tvorbě tabulky prodeje produktů evidujeme název produktu, jenž je textového typu, datum spotřeby datového typu a počet prodaných kusů, jenž bude nabývat číselných hodnot, a proto bude numerického typu. [7]

Datové typy podporované SQL Serverem 2008: Binary, bit, char a varchar, date, datetime a datetime2, datetimeoffset, decimal, float a real, int, bigint, smallint, tinyint, money a smallmoney, nchat a nvarchar, smalldatetime, time. [12]

2.5.2 Vybrané příkazy jazyka SQL

SQL obsahuje velkou množinu příkazů, proto jich uvedu jen několik, podstatných především pro vlastní návrh.

Budování databáze:

- CREATE TABLE

Příkaz pro vytvoření tabulky. Obsahuje atributy s určeným datovým typem a jeho případným datovým omezením. Dále je zde uveden primární klíč.

- DROP

Příkaz pro smazání. Po smazání se ztratí i veškerá data.

Ukázka: DROP TABLE XY - příkaz pro smazání tabulky XY

- ALTER

Příkaz pro upravení. V jistém smyslu lepší než DROP, například v případě přidání řádku do tabulky je lepší použit příkaz ALTER, než smazat tabulku a znovu ji vytvořit s novou strukturou.

Ukázka: ALTER TABLE XY - upraví strukturu tabulky.

- TRIGGER

Spouštěč neboli trigger je procedura, která se spustí v okamžiku, kdy v zadané tabulce dojde k nějaké události. Události spouštěče mohou na zadaných databázových tabulkách provádět příkazy, jako je INSERT, UPDATE a DELETE. Spouštěče spouští implicitně databázový server, jakmile na sledované tabulce dojde k nějaké události. Bez ohledu na identitu uživatele se tedy Trigger používá především pro manipulaci s daty a zajištění předem daných pravidel. Spouštěč je tvořen ze tří částí: prováděcího příkazu, omezení a akce spouštěče. [15]

Operace se záznamy:

- INSERT INTO

Příkaz důležitý pro naplnění tabulky daty. Technicky přidává nový řádek.

Ukázka: INSERT INTO Knihy (název, kategorie) VALUES (‘QUO VADIS‘,‘román‘)

- UPDATE

Upravuje záznam (řádek) tabulky.

- DELETE

Smaže záznam (řádek) tabulky

Náhled na data:

- SELECT

Důležitý příkaz i pro další tvorbu kódu, například pro vytvoření pohledů (CREATE VIEW). Pomocí toho příkazu můžeme z databáze dostat určitou množinu dat. Ukázka: SELECT (jméno, příjmení) FROM Zaměstnanci (ukáže všechny hodnoty atributů jména a příjmení z tabulky Zaměstnanci).

- VIEW

Pohled je uložený databázový dotaz, který uživateli poskytuje přizpůsobenou podmnožinu dat z jedné nebo několika databázových tabulek. Lze jej vytvořit pomocí příkazu CREATE VIEW. [14]

2.6 Index

Index představuje způsob prezentace dat odlišným způsobem, než jak vypadá na disku. Speciální typy indexů přeuspořádají fyzické umístění záznamu v tabulce. Indexy lze vytvářet na sloupci v tabulce nebo na kombinaci sloupců v tabulce. Když se používá index, data se uživateli prezentují v uspořádaném pořadí, které lze řídit pomocí příkazu CREATE

INDEX. Značného nárůstu výkonu lze dosáhnout indexováním správných polí, především pak polí, která se používají ke spojování tabulek. [15]

Z těchto důvodů jsou indexy důležité i pro tuto práci. Doba potřebná pro vyhledání dané informace má být co nejkratší a lze toho dosáhnout správnou indexací správných polí jednotlivých tabulek.

2.6.1 Clusterované indexy

Clusterovaný index je výjimečný tím, že díky němu systém SQL Server uspořádá data v tabulce podle clusterovaného klíče. Vzhledem k tomu, že tabulku nelze seřadit více způsoby, je možné v jedné tabulce definovat jen jediný clusterovaný index. Výchozí nastavení primárního klíče má hodnotu „clustered“, proto systém SQL Server vytvoří pro primární klíč clusterovaný index. Obdobně platí, že omezení UNIQUE je fyzicky implementováno jako jedinečný index. [6]

2.6.2 Neclusterované indexy

Neclusterované indexy nevynucují určité řazení tabulky, takže jich v jedné tabulce může být i více. Indexy mohou zrychlit provádění dotazů, ale každý vytvořený index zároveň zhoršuje odezvu všech operací INSERT, UPDATE, DELETE a MERGE. Proto musí být počet indexů optimální. [6]

3 Popis současné situace

Popis současné situace je rozdělen do několika částí. Lze na ně nahlížet jako na části, jež jsou detailně zpracovány a jsou nutné pro adekvátní zhodnocení celkové situace. Kupříkladu tak můžeme nahlížet na jednotlivé akademické pracovníky a jejich osobní údaje. Na druhou stranu je zpracován vztah tohoto pracovníka k projektové skupině. Vzniká tak tedy i popis vazeb mezi jednotlivými částmi.

3.1 Zaměstnanci

Vysoká škola vede, zejména pro účely zjišťování skutkového stavu v řízeních ve věcech akreditací, v elektronické podobě registr docentů a profesorů zaměstnaných na veřejných a soukromých vysokých školách, obsahující o uvedených zaměstnancích zejména tyto údaje: osobní údaje (jméno, příjmení, rodné číslo, datum narození a místo trvalého pobytu), údaje o vzdělání (udělené tituly, obor vzdělání, vědecké hodnosti), údaje o pracovním poměru (vznik, začátek a změny pracovního poměru, včetně údajů o rozsahu práce) a údaje o pracovním zařazení jako docent nebo profesor. [3]

Na vysokých školách se vyskytují i další osoby, jež musí být evidovány v systému kvůli potřebě pokrytí celé struktury. Tyto osoby mají stejné vlastnosti osobních informací a údajů o vzdělání. Proto jim náleží i stejná struktura.

Zároveň by tato část měla souhrnně zajišťovat pohledy za účelem vyhledávání zaměstnanců. Je žádoucí koncipovat konečný výstup ve formě přehledů o informacích vyskytujících se především v životopisech jednotlivých zaměstnanců.

3.2 Předměty

Vysoká škola uskutečňuje akreditované studijní programy a programy celoživotního vzdělávání. Typ vysokoškolské vzdělávací činnosti je určen typem uskutečňovaných akreditovaných studijních programů. Tyto typy studijních programů jsou bakalářský,

magisterský a doktorský. Přičemž součástmi studijního programu jsou: název studijního programu, jeho typ, forma a cíle studia. [3]

Požadavkem, kromě evidence jednotlivých předmětů formou historie výuky a jejich vyučujících, je i určení přibližné zatíženosti a počtu odučených hodin jednotlivých učitelů v rámci výuky. Otevře se tak možnost pro lepší plánování a usměrnění jednotlivých pedagogů.

3.3 Publikace

Publikace mají být evidovány, a to jak základní atributy (název, abstrakt, …), tak i například napojení na projekty, v jejichž rámci byla práce publikována. Z hlediska monitoringu dále odkazuje na souhrnné údaje týkající se udělených bodů za publikace v rámci jednotlivých fakult.

Má tedy pojednávat o typu publikace a počtu dosažených bodů v rámci státní databáze. Nabízí se zde další větvení typu dle specializace. V případě ekonomického větvení by primární rozdělení měla být určena dle struktury dané Radou pro vědu a výzkum a její bližší specifikace v oblasti ekonomie za pomoci struktury JEL.

Klasifikační systém JEL byl vyvinut za účelem použití v *Journal of Economic Literature* (*JEL*) a je standardem metod a klasifikací školní literatury na poli ekonomie. Systém je použit na klasifikování článků, disertačních prací, knih, recenzí, slohových prací v oblasti ekonomické literatury a v mnoha dalších případech. [1]

3.4 Projekty

Vysoké školy jako nejvyšší článek vzdělávací soustavy jsou vrcholnými centry vzdělanosti, nezávislého poznání a tvůrčí činnosti a mají klíčovou úlohu ve vědeckém, kulturním, sociálním a ekonomickém rozvoji společnosti tím, že uchovávají a zvětšují dosažené poznání a podle svého typu a zaměření pěstují činnost vědeckou, výzkumnou, vývojovou a

inovační, uměleckou nebo jinou tvůrčí činnost. Dále pak rozvíjejí mezinárodní a zvláště evropskou spolupráci jako podstatný rozměr svých činností, podporují společné projekty s obdobnými institucemi v zahraničí, vzájemné uznávání studia a diplomů či výměnu akademických pracovníků a studentů. [3]

V další řadě je důležité evidovat zobrazení národních a mezinárodních projektů. Poté roztřídění podle oborů a zobrazení členů s jejich funkcemi v rámci projektové části. Výstupem projektu může být publikace, a proto zde musí existovat i vazba mezi těmito entitami.

3.5 Závěrečné práce

Vyžaduje se souhrnný počet závěrečných prací v rámci vedoucích práce a typu práce. Dále evidování uděleného hodnocení, a zda byla práce obhájena či nikoliv.

4 Analýza problému

Z rozboru zkoumaného předmětu či situace jednotlivých částí vyplývá podrobnější poznání jednotlivých jevů, které také usnadňují poznání celkové struktury. Kupříkladu z hlediska vytíženosti jednotlivých vyučujících je vhodné rozprostřít časové nároky mezi jednotlivé kantory. Předejde se tím problémům, jako je vysoce zaneprázdněný vyučující a naopak učitel, jenž by si rád přibral hodinu navíc v rámci svého pole působnosti.

Dalším problémem je, že práci vyučujících často ztěžuje vyhledávání a vyhodnocování potřebných dat. Údaje, které jsou papírově podloženy a je třeba je kolikrát zdlouhavě dohledávat. Elektronická forma disponuje mnohem rychlejšími možnostmi, jak účinně vyhledávat. Proto vznikl návrh na vytvoření databázového systému, který bude vyhovovat různým typům jednotlivých úkonů.

Uživatelé mají tedy získat přehled nejen o svých vlastních činnostech a spolupracovnících, ale i celofakultní pohled na ostatní uživatele v databázi. Tento systém ovšem bude fungovat jen za předpokladu, že uživatelé budou ochotni poskytovat jednotlivé informace a pravidelně je aktualizovat.

Specifikace databázového systému je rozdělena do několika oblastí, z nichž uvedu jen ty nejvýznamnější.

4.1 Zaměstnanci

Evidence zaměstnanců má podporovat třídění v rámci vykonávaných funkcí, znalosti jednotlivých jazyků a pedagogické a pracovní zkušenosti. Výstup je zde vhodné prezentovat v podobné formě jako například strukturovaný životopis. Demonstruje se tím funkčnost a použitelnost v praxi.

4.2 Předměty

Z hlediska podpory při plánování a realizaci výuky se očekávají výstupy v podobě pohledů. Tyto pohledy budou podávat souhrnné informace ohledně monitoringu jednotlivých vyučujících v rámci odučených hodin, profesního růstu a jiné. Garanti předmětů uvítají zobrazení počtu nutných hodin vzhledem k počtu studentů a kapacitě učeben, vytíženosti jednotlivých vyučujících vzhledem ke všem jejich profesním činnostem a další.

4.3 Projekty

Pohled na projekty má přinášet seznam projektů v rámci skupin a jejich členů. Bližší pohled pak uvádí podrobnější specifikaci a kategorizaci projektu. Výstupem projektu může být publikační činnost.

4.4 Publikace

Předmětem výsledku části o publikační činnosti je především seznam publikací jednotlivých zaměstnanců. V tomto seznamu lze vyhledávat za pomoci kategorizace oborů, která je daná strukturou podle Rady pro vědu a výzkum. Třídění nabízí i bližší specifikaci v oblasti ekonomie podle struktury JEL.

V rámci jednotlivých ústavů a fakult jsou zde pohledy, které poskytují souhrnné údaje o počtu získaných bodů za publikace v jejich rámci. Další pohledy poskytují náhled na počet publikací a získaných bodů v rámci jednotlivých zaměstnanců.

4.5 Problémy v akademickém prostředí

V dané oblasti se vyskytuje mnoho problémů, jež je třeba řešit. Ukázkově nastíním pár problematických příkladů, jejichž řešení je zahrnuto a vyřešeno v tomto systému.

4.5.1 Hledání spolupracovníků na projekt

Příklad: Pro daný projekt jsou vyžadování lidé, již jsou ze stejné fakulty a mají jisté specifické požadavky. Konkrétně znalost anglického jazyka na velice dobré úrovni.

Uživatel si tedy vyfiltruje všechny uživatele, již odpovídají daným podmínkám v pohledu zkušeností. Následně si blíže zobrazí vybrané kandidáty a ty poté kontaktuje na základě jimi zvolených kontaktních údajů, případně pošle přímo pozvánku.

4.5.2 Vedení projektu

Příklad: Uživatel dostal pozvánku na projekt, ale nezná podrobné informace. Navíc by rád zjistil, kdo daný projekt vede a kteří lidé na něm participují.

Na základě pozvánky si v projektovém pohledu vyhledá daný projekt, kde zjistí veškeré podstatné informace, včetně členů a jejich funkce.

4.6 Výběr nástrojů

Vzhledem k dostupnosti, kompatibilitě a použitelnosti jsou vybrány následující nástroje pro tvorbu databázového systému.

4.6.1 SQL Server

Pod označením SQL Server se neskrývá pouze databáze, ale komplexní moderní, výkonná, spolehlivá a bezpečná serverová platforma pro ukládání a správu údajů v databázích a datových skladech a sady nástrojů pro Business Inteligence včetně pokročilého reportování. Uplatní se v široké škále podniků všech velikostí. Poradí si nejen s relačními a multidimenzionálními údaji, ale poskytne spolehlivé úložiště i pro multidimenzionální dokumenty a soubory. [10]

SQL server je brán jako jeden ze standardů, a proto je zvolen i pro tuto práci. Nicméně není velkým problémem spustit tuto databázi i na jiné platformě za pomoci drobných úprav. Stejně byla zvolena i verze systému 2008, přestože během tvorby byla vydána verze 2014. Verze 2008 je prozatím dostupnější a při upgradu na vyšší verzi by měla být zachována kompatibilita návrhu s touto verzí.

4.6.2 SQL Server Management Studio

Univerzální a komplexní nástroj pro účely formulování SQL příkazů. Slouží pro odevzdání příkazu databázovému serveru a usnadňuje zjištění, zdali je příkaz syntakticky správně, nebo zda vykonal, co jsme požadovali. [10]

SQL Server Management Studio bylo vybráno jako hlavní nástroj pro tvorbu databázového systému. To především z již zmíněných důvodů korekce syntaktických chyb a možnosti odevzdávat příkazy databázovému serveru. Nicméně pro generování schémat jsem zvolil dbForge Studio for SQL Server, a to především z důvodu estetické stránky věci.

5 Vlastní návrhy řešení

Tato kapitola obsahuje stěžejní část práce, pojednávající nejen o návrhu jednotlivých entit a atributů důležitých pro vznik databázového systému. Postupně se přidávají další entity a atributy pro smysluplnější chod databáze. Nad tabulkami jsou vytvořeny pohledy zajišťující přirozenější pohled na uložená data. Veškerý chod je zajištěn za pomoci jednotlivých procedur a funkcí, které mimo jiné ošetřují data, jež vstupují do databáze.

5.1 Návrh

Extrakce samotných požadavků na systém přináší i primární požadavky na návrh entit a atributů.

Řešení bude probíhat na bázi MS SQL Serveru. Důvodů, proč jsem zvolil MS SQL Server, je hned několik. V mnoha situacích nabízí vysoký výkon a také podporu pro velké databáze. Tyto databáze mohou být zálohované i během používání jinými klienty. V případě selhání systému je možnost využít automatického obnovení z těchto záloh bez nutnosti zásahu administrátora. Pokud se podíváme na vývoj do budoucna, tak je tento produkt vyvíjen společností Microsoft a lze předpokládat snazší využití v dalších programech od MS jako je například Access.

5.1.1 Zaměstnanci

U zaměstnanců evidujeme základní údaje (jméno, příjmení, rodné číslo, …). Tuto množinu je třeba rozšířit o další údaje, jako jsou například:

- Vystudované školy

Podle ukončení daných škol a univerzit se automaticky evidují i tituly a jsou následně zobrazeny i v hlavičce uživatele, pokud si to uživatel přeje.

- Telefony

Jeden zaměstnanec má více telefonů, u kterých může evidovat účel telefonu. Přičemž jeden telefon náleží jen jednomu zaměstnanci. Výjimkou jsou telefony umístěné v kabinetech. Tyto telefony uživatel nezapisuje. Jsou mu automaticky zobrazeny po uvedení své kanceláře. Předvolba telefonního čísla je zobrazena automaticky dle daného státu.

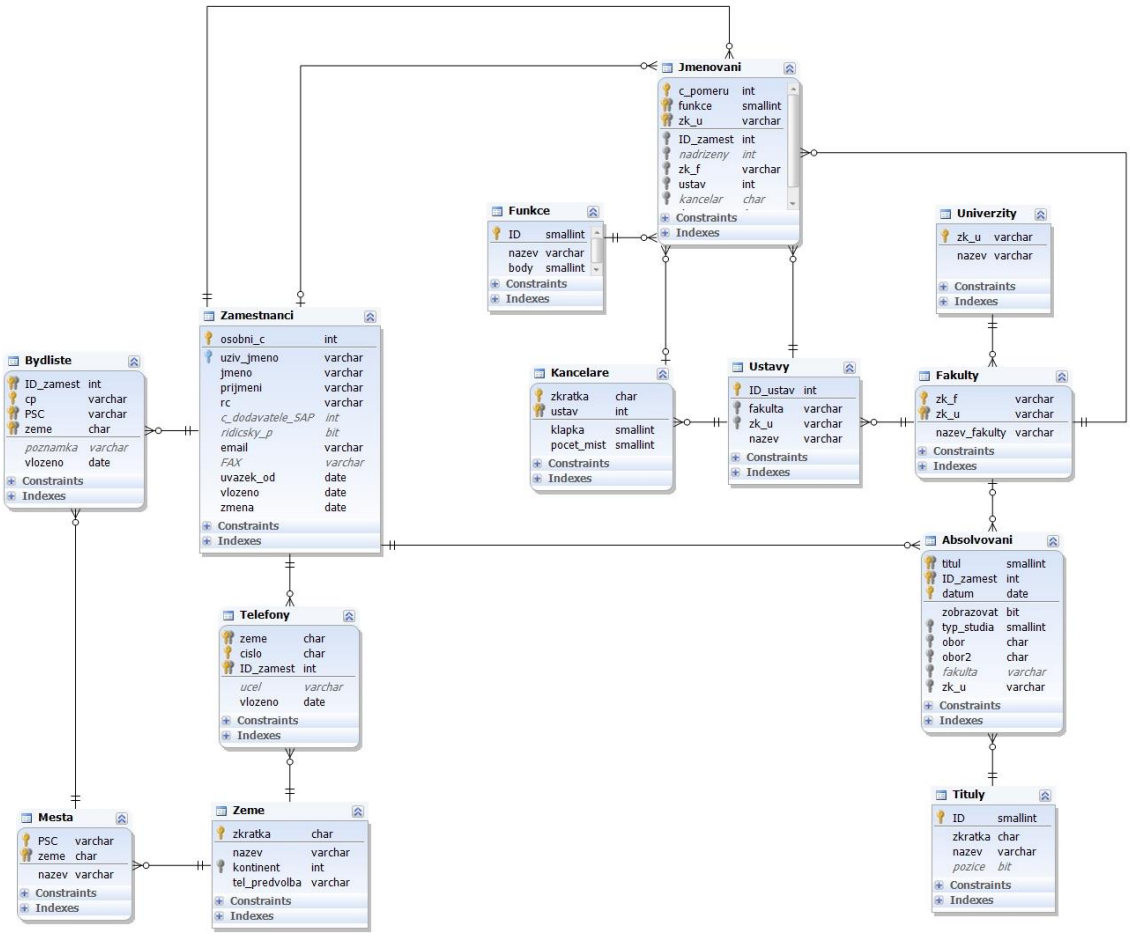
- Oblast působení

Uvádí, na kterém ústavu, které fakulty zaměstnanec pracuje. Toto je evidováno skrze jmenování do funkce. Proto je uvedeno nejen, jaké funkce zaměstnanec vykonává, ale i kde je vykonává.

- Bydliště

Uživatel může evidovat více svých adres, přičemž může přidat poznámku například o specifikaci účelu této adresy.

Obrázek 1 – Grafická ukázka schématu zaměstnanců



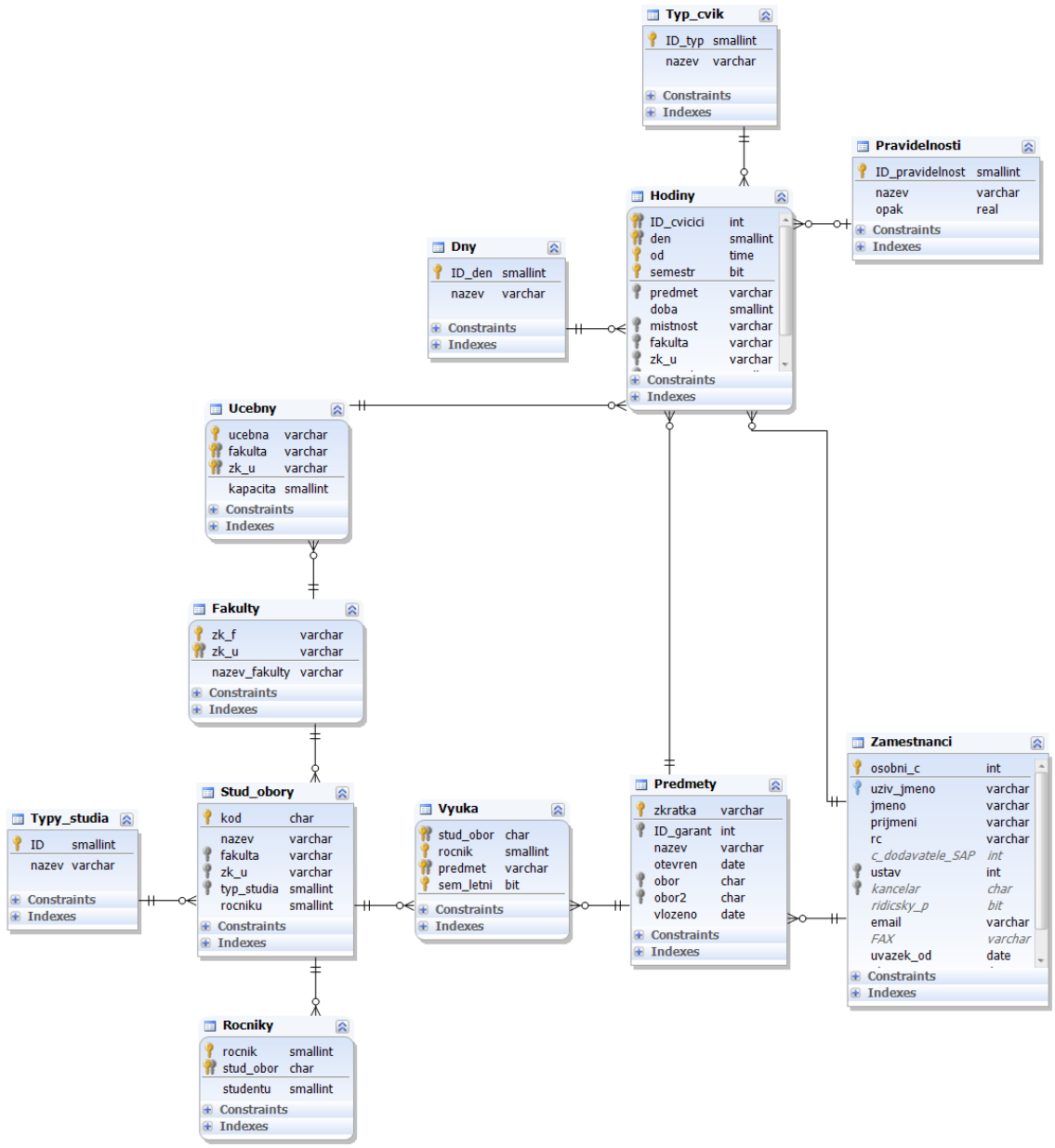
Zdroj: vlastní zpracování

5.1.2 Předměty

Evidence předmětů je důležitá především z hlediska ukládání základních informací, jako je například evidence garantů jednotlivých předmětů v průběhu let. Dále pak má tato část pomoci při plánování výuky.

- Předměty
- Hodiny
- Místo výuky

Obrázek 2 – Grafická ukázka schématu předmětů



Zdroj: vlastní zpracování

5.1.3 Projekty

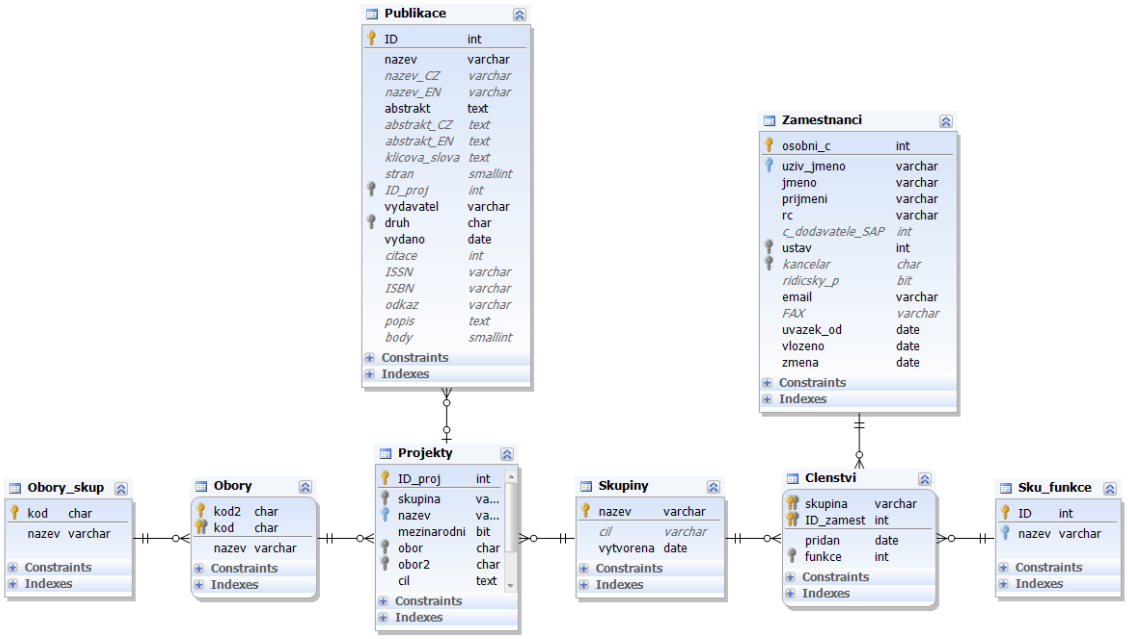
- Projekty

U projektů jsou evidovány jak základní údaje, jako jsou název či cíl, tak i oblast, ve které je většinově obsažen. Výstupem projektu může být publikace.

- Skupiny

Jednotlivé projekty jsou vypracovávány ve skupinách, jež jsou zodpovědné za jednotlivé projekty. Pro rozložení vazby M:N se zaměstnanci je vytvořena tabulka členství. V této tabulce lze definovat jednotlivé funkce pedagogického pracovníka v dané skupině. Toho je dosaženo za pomoci číselníku, jejž reprezentuje tabulka funkcí (Sku_funkce).

Obrázek 3 – Grafická ukázka schématu projektů

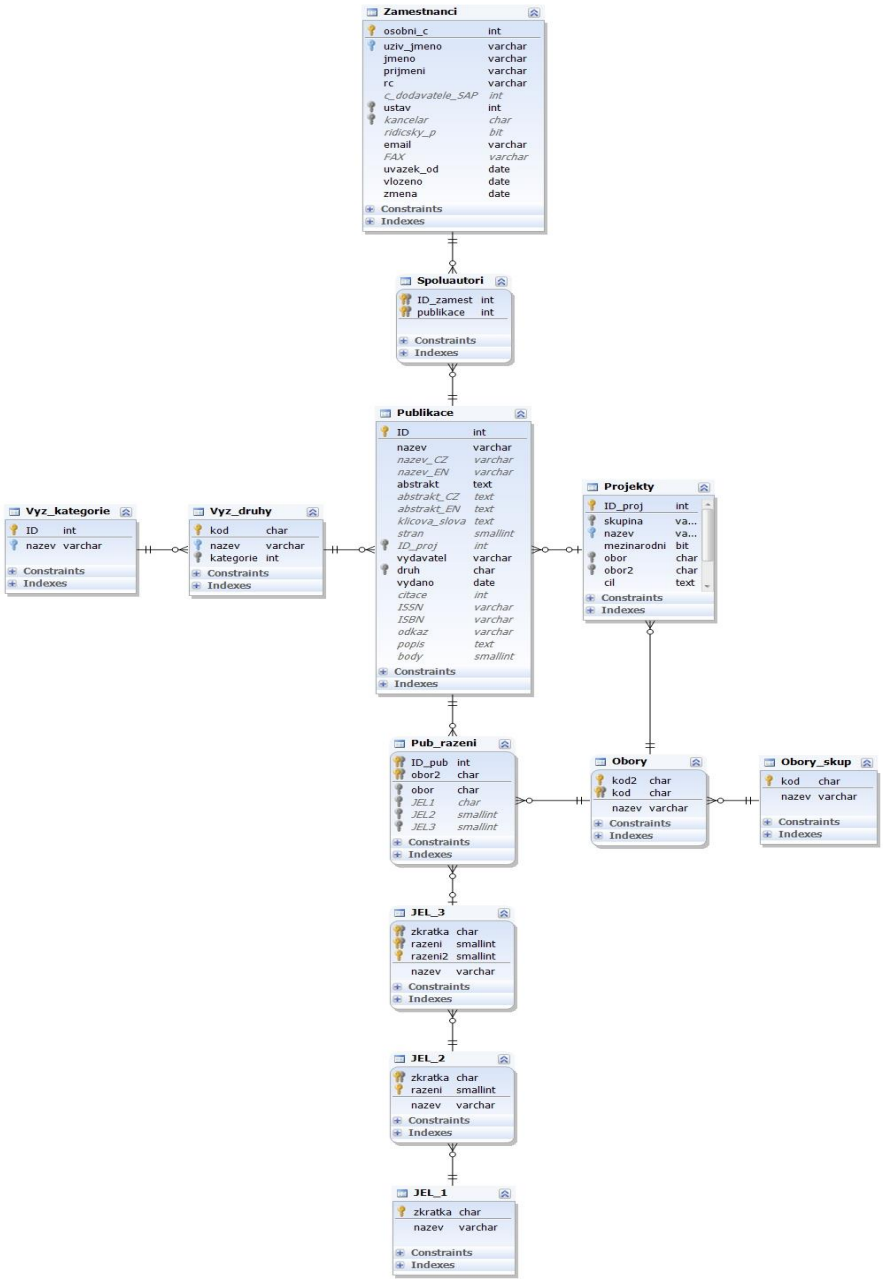


Zdroj: vlastní zpracování

5.1.4 Publikace

Oblast publikací je značně rozvětvená především z hlediska řazení podle druhu výzkumu a seznamu oborů, v nichž se vyskytuje. Oblast ekonomie je blíže specifikována za pomoci struktury JEL (Journal of Economic Literature).

Obrázek 4 – Grafická ukázka schématu publikací



Zdroj: vlastní zpracování

5.1.5 Certifikace

Slouží k evidenci dosažení jednotlivých certifikátů a jazykových zkoušek v rámci jednotlivých pracovníků.

- Certifikáty

  Zaznamenáváme důležité certifikáty pracovníků, jež jsou tříděny dle oborů. Velkou částí se předpokládají certifikáty z jazykových zkoušek, proto mají vytvořenu i svoji specifickou strukturu.

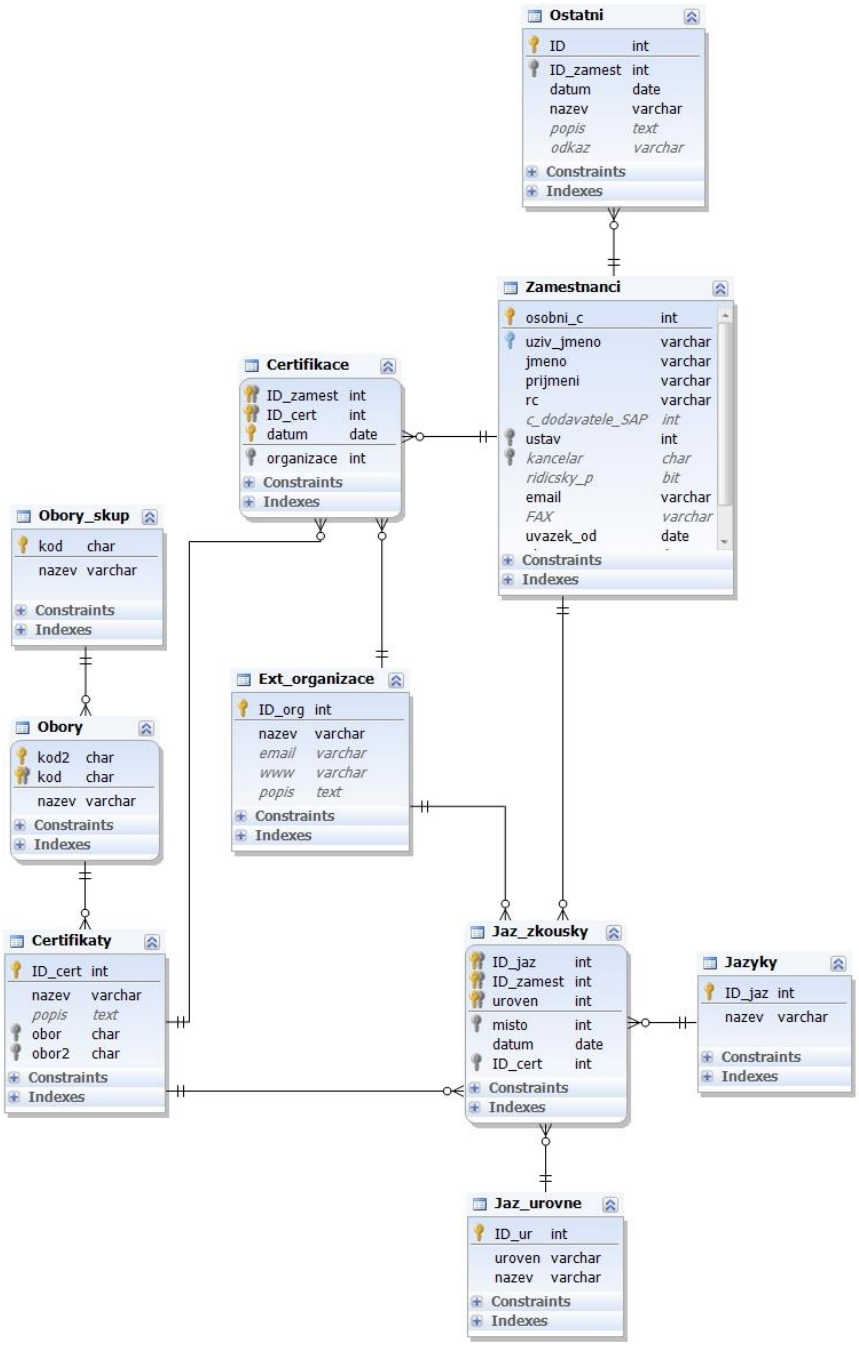
- Jazykové zkoušky

  Dle rozdělení na jednotlivé jazyky a úrovně vykonaných zkoušek jsme schopni kontrolovat jazykové znalosti v rámci fakulty či jednotlivců.

- Ostatní

  Pro zachycení veškerých dalších důležitých milníků ze života uživatele slouží tabulka ostatní. Slouží především pro výpis životopisu, ať se jedná třeba o dosažení ceny ve svém oboru nebo jiné úspěchy skupiny či jedince.

Obrázek 5 – Grafická ukázka schématu certifikací

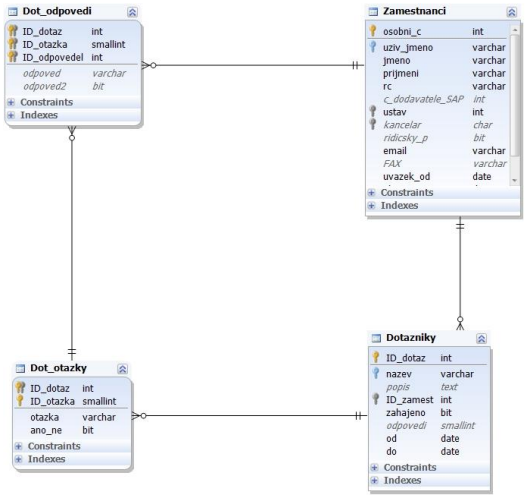


Zdroj: vlastní zpracování

5.1.6 Dotazníky

Možnost oslovit ostatní pracovníky, případně oslovit administrátora, zajistí tato struktura. Za pomoci pohledů je poté vytvořen přehled o jednotlivých dotaznících a jejich otázkách a odpovědích.

Obrázek 6 – Grafická ukázka schématu dotazníků



Zdroj: vlastní zpracování

5.1.7 Zkušenosti

V rámci nabytých zkušeností je možno dosažené znalosti rozdělit do tří hlavních skupin, a to podle pedagogické či pracovní činnosti a zkušenosti nabyté studiem.

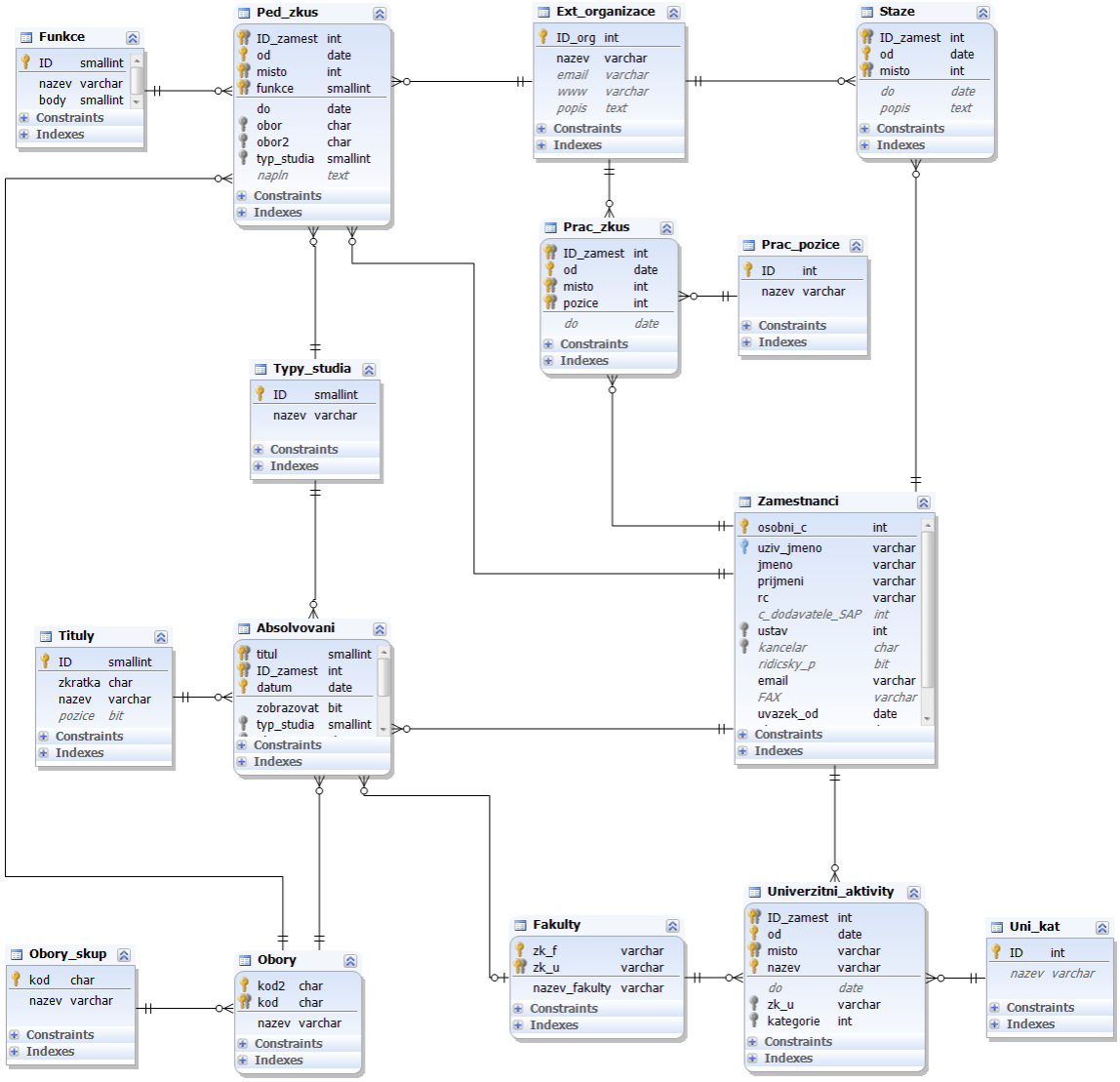
Pedagogická činnost zahrnuje přehled dosažených vědomostí v rámci pedagogického působení v různorodých institucích. Zabývá se především evidencí lokalit, kde daný pedagog působil, ve kterých letech a jakou funkci zde vykonával. Funkce jsou pro uživatele předdefinovány a obsahují známé funkce ve školství. Pro lepší orientaci je pedagogické působení rozděleno dle jednotlivých oborů, do kterých náleží. Tato struktura je dána podle struktury Odboru Rady pro výzkum, vývoj a inovaci. Specialitou v pedagogické činnosti

jsou univerzitní aktivity. Jedná se především o evidenci členství v různorodých uskupeních vykonávaných i na různých univerzitách.

Z hlediska minulého a současného zaměstnání je vytvořena struktura poskytující přehled o časovém úseku, kdy se uživatel nacházel v daném zaměstnání a na které pozici.

Studijní vědomosti primárně zaznamenávají, kde, v kterém roce a v jakém oboru daný uživatel absolvoval. Nicméně vedlejší funkcí zde je přidělení titulu podle dosaženého vzdělání a následné automatické umístění ke jménu. Tituly jsou předdefinované a obsahují veškeré české tituly i známější zahraniční tituly.

Obrázek 7 – Grafická ukázka schématu zkušeností



Zdroj: vlastní zpracování

5.2 Archivace a zálohování

Z hlediska potřeby evidovat zrušené předměty a bývalé pracovníky vzniká potřeba tyto údaje dlouhodobě ukládat. Evidují se zde základní informace, jež jsou automaticky zachovány při mazání dat z klasické struktury, a to za pomoci vytvořených triggerů.

5.3 Tvorba tabulek

Ukázka vytvoření jedné z mnoha tabulek. Konkrétně tabulky zaměstnanců. Prvně by si pracovník měl zvolit, případně mu bude vygenerováno jeho uživatelské jméno. Následně vyplní jméno, příjmení, rodné číslo a ostatní. Primárním klíčem jsem zvolil osobní číslo. Dalšími kandidátními klíči by mohlo být například uživatelské jméno nebo rodné číslo. Ostatní údaje o zaměstnanci jsou uloženy v dalších tabulkách.

```
CREATE TABLE Zamestnanci
(
        uziv_jmeno VARCHAR(20)NOTNULLUNIQUE,
        jmeno VARCHAR (20)NOTNULL,
        prijmeni VARCHAR (20)NOTNULL,
        osobni_c INTNOTNULL,
        c_dodavatele_SAP INT,
        ridicsky_p BIT,
        email VARCHAR (50)NOTNULL,
        FAX VARCHAR (9),
        vlozeno DATENOTNULLDEFAULTGETDATE(),
        zmena DATENOTNULLDEFAULTGETDATE(),
        PRIMARYKEY (osobni_c)
)
GO
```

5.4 Funkce, Procedury

Samotný chod databáze je veden za pomoci procedur a funkcí. Ty zajišťují různé aspekty, jež umožňují plynulý chod systému. Ukázky jednotlivých funkcí a procedur jsou uvedeny v podkapitolách. Další můžete nalézt na přiloženém CD.

5.4.1 Procedury ošetřující vstupy od uživatelů

Je důležité ochránit vstupy databáze nejenom před SQL Injection, ale například i před náhodnými chybami uživatelů. Vstup do databáze je proto běžnému uživateli umožněn pouze přes tyto procedury. Procedury tak obsahují kontrolu jednotlivých vstupů, jakou jsou například kontrola tvaru zadaného e-mailu, kontrola data narození a podobně.

Ukázka procedury pro vložení externí organizace:

```
CREATE PROCEDURE MY_vloz_organizace
@nazev VARCHAR (70),
@email VARCHAR (50),
@www VARCHAR (50),
@popis TEXT
AS
BEGIN
	IF @email isNOTNULLAND @email NOTLIKE'%_@_%_.__%'
	PRINT('neplatný email')
	ELSE
		IF @www isNOTNULLAND @www NOTLIKE'%_.__%'
			PRINT('neplatné webové stránky. vzor:(www.seznam.cz)')
		ELSE
		BEGIN
			INSERTINTO Ext_organizace VALUES (@nazev,@email,@www,@popis)
			PRINT ('organizace vložena')
		END
END
GO
```

5.4.2 Funkce pro formátování textu

Pro ochranu před chybami uživatelů jsem vytvořil funkci, jež převede vložený text na text s malými písmeny a velkými počátečními. Funkce je použita v procedurách pro kontrolu vkládání jednotlivých jmen a příjmení.

```
CREATE FUNCTION MY_jmeno

(@text VARCHAR(100))

RETURNSVARCHAR(100)

AS

BEGIN

DECLARE @navrat VARCHAR(100)=UPPER(LEFT(@text,1))+LOWER(RIGHT(@text,LEN(@text)-1))

RETURN @navrat

END

GO
```

5.4.3 Procedura pro výpis životopisu zaměstnance

Obsahuje především kurzory, jež vypíší z jednotlivých tabulek podstatné informace o pracovníkovi. Výstupem je zde forma životopisu poskytující základní údaje o zaměstnanci, bydliště, telefony, historie vzdělání, pedagogické a pracovní činnosti a další podstatné údaje. Kompletní kód je velmi obsáhlý a je proto k dispozici na přiloženém CD.

Ukázka části výpisu z procedury po zadání příkazu EXEC MY_ziv @ID_in=222, jež spustí výpis životopisu pro uživatele s ID 222.

1. Osobní informace

Osobní číslo: 222

Jmenovka: Bc. Ing. PhDr. RNDr. Vladimír Skácel, Ph.D., Th.D.

Bydliště: Krátká 1192 Brno 12

E-mail: lada2@seznam.cz

Telefon: +33 707 112 345 pracovní

Telefon2: +420 455 112 345 služební

Telefon3: +420 605 488 222

2. pedagogická činnost

obor	**od**	**do**	**Univerzita**	**funkce**
Informatika	2005	2010	VYSOKÉ UČENÍ TECHNICKÉ	hostující profesor
Informatika	2009	2011	VYSOKÉ UČENÍ TECHNICKÉ	kancléř
Informatika	2012	2013	VYSOKÉ UČENÍ TECHNICKÉ	kancléř

3. Pracovní zkušenosti

od	**do**	**místo**	**pozice**
2005	2009	Dopráce s.r.o	technik
2008	2010	Certifikace a.s.	technik
2010	****	Certifikace a.s.	manažer

5.5 Triggery

Především pro zápis do datových skladů je důležité vytvořit automatizovaný převod těchto dat z klasické struktury do struktury datových skladů. Většina těchto triggerů funguje při mazání a úpravě dat v původní struktuře.

Ukázka části triggeru, který při smazání dat z tabulky Zaměstnanci převede tato data do struktury datových skladů. Navíc vyvolá příkaz na smazání dat z provázaných tabulek.

```
CREATE TRIGGER SMAZ_zamestnance ON Zamestnanci

INSTEAD OF DELETE

AS

    BEGIN

        DELETE FROM Bydliste WHERE Bydliste.ID_zamest=(SELECT
        osobni_c FROM DELETED)

        DELETE FROM Ped_zkus WHERE Ped_zkus.ID_zamest=(SELECT
        osobni_c FROM DELETED)

        DELETE FROM Staze WHERE Staze.ID_zamest=(SELECT osobni_c
        FROM DELETED)

        DELETE FROM Absolvovani WHERE Absolvovani.ID_zamest=(SELECT
        osobni_c FROM DELETED)

        DELETE FROM Hodiny WHERE Hodiny.ID_cvicici=(SELECT osobni_c
        FROM DELETED)

        DELETE FROM Clenstvi WHERE ID_zamest=(SELECT osobni_c FROM
        DELETED)
```

```
        UPDATE Zamestnanci SET smazano=1 WHERE osobni_c=(SELECT
        osobni_c FROM DELETED)

        UPDATE Zamestnanci SET smazano_d=GETDATE() WHERE
        osobni_c=(SELECT osobni_c FROM DELETED)

    END

GO
```

5.6 Pohledy

Pohledy slouží uživateli především pro lepší pohled na data v databázi. Jsou zde jak souhrnné pohledy pro celofakultní přehled, tak i jednotlivé uživatelské pohledy. Ukázky jednotlivých pohledů jsou níže, další můžete nalézt na přiloženém CD.

5.6.1 Zaměstnanci

Přehled základních údajů o jednotlivých zaměstnancích. Zde se projevuje přidělení jednotlivých titulů danému jedinci a zobrazení základních údajů. Další rozšiřující informace jsou poskytnuty v rámci dalších pohledů.

```
CREATEVIEW poh_Zamestnanci AS

(

    SELECT

    STUFF(

        (SELECTRTRIM(' '+ pro0.titul)

        FROM proces_zamestnanec0 pro0

        WHERE pro0.ID = Zam.osobni_c
```

```
            FORXMLPATH (''))
            , 1, 1,'')AS'Titul pred',
        Zam.jmeno, Zam.prijmeni,
        STUFF(
            (SELECTRTRIM(' '+pro.titul)
            FROM proces_zamestnanec pro
            WHERE pro.ID = Zam.osobni_c
            FORXMLPATH (''))
            , 1, 1,'')AS'Titul za',
Zam.osobni_c, Zam.c_dodavatele_SAP, Zam.ridicsky_p, Zam.email, Zam.uvazek_od
        FROM Zamestnanci Zam
GROUPBY Zam.osobni_c, Zam.jmeno, Zam.prijmeni, Zam.c_dodavatele_SAP,
Zam.ridicsky_p, Zam.email, Zam.uvazek_od
)
GO
```

Tabulka 1: Výpis z poh_Zamestnanci

Titul pred	jmeno	prijmeni	Titul za	osobni_c	email
NULL	Ondřej	Skoupil	NULL	123	SkoupilOndrej@seznam.cz
Ph.D. Th.D.	Vladimír	Skácel	Bc. Ing. PhDr. RNDr.	222	lada2@seznam.cz
NULL	Petr	Doležal	NULL	333	petr007@seznam.cz
Th.D.	Vanda	Miloševský	PhDr. RNDr.	555	milos@seznam.cz

Zdroj: vlastní zpracování

5.6.2 Předměty

Jednotlivé pohledy podávají přehlednější informace o předmětech ve více úrovních. V primární úrovni se jedná o vývoj předmětu v průběhu let. Sekundární část poskytuje důležité informace pro plánování výuky a pro jednotlivé garanty. Terciární je nakonec zaměřena na jednotlivé vyučující a vyučované hodiny.

Ukázka z terciární úrovně:

```
CREATEVIEW poh_predmety_vyucujici AS

(

    SELECT Zam.jmeno, Zam.prijmeni, Pre.zkratka, Pre.nazev AS'předmět', tp.nazev AS'typ',LEFT(Hod.od,5)AS'zacatek',


    (LEFT((CAST(LEFT(Hod.od,2)ASINT)+Hod.doba),2)+RIGHT(LEFT(Hod.od,5),3 ))AS'konec',Hod.doba AS'hodin', Hod.mistnost AS'učebna'

    FROM Hodiny Hod

    LEFTJOIN Predmety pre
```

```
        ON pre.zkratka=Hod.predmet
        RIGHTJOIN Zamestnanci Zam
        ON Zam.osobni_c=Hod.ID_cvicici
        LEFTJOIN Typ_cvik tp
        ON tp.ID_typ=Hod.typ_cvika
        LEFTJOIN Pravidelnosti pr
        ON pr.ID_pravidelnost=Hod.pravidelnost
)
GO
```

Tabulka 2 – Ukázka výpisu z databáze - předměty

jméno	příjmení	zkratka	předmět	typ	začátek	konec	hodin	učebna
Petr	Skoupil	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Vladimír	Skácel	ZP	Základy podnikání	cvičení	7:00	9:00	2	P285
Vladimír	Skácel	ZP	Základy podnikání	cvičení	9:00	11:00	2	P285
Vladimír	Skácel	ZP	Základy podnikání	cvičení	11:00	13:00	2	P285
Vladimír	Skácel	ZP	Základy podnikání	přednáška	11:00	14:00	3	P485
Petr	Doležal	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Vanda	Milešovský	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Zdroj: vlastní zpracování

5.6.3 Projekty

Souhrnný pohled ukazující detailní náhled jak na téma projektu, tak i na to, která skupina jej vykonává a na další náležitosti.

```
CREATE VIEW poh_Projekty AS

(

    SELECT Sk.nazev AS'Skupina', Pr.ID_proj AS'ID projektu', Pr.nazev, Pr.cil, Pr.popis, Pr.zahajeni, Pr.ukonceni,
```

```
Fa.nazev_fakulty AS'fakulta', Us.nazev AS'ústav', Pr.mezinarodni AS'mezinárodní', os.nazev AS'kategorie', ob.nazev AS'obor'
        FROM Projekty Pr
        INNERJOIN Skupiny Sk
        ON Sk.nazev=Pr.skupina
        INNERJOIN Clenstvi Cl
        ON Cl.skupina=Sk.nazev
        LEFTJOIN Jmenovani jme
        ON Jme.ID_zamest=Cl.ID_zamest
        LEFTJOIN Ustavy Us
        ON us.ID_ustav=jme.ustav
        INNERJOIN Fakulty Fa
        ON Fa.zk_f=Us.fakulta
        LEFTJOIN Obory ob
        ON ob.kod=Pr.obor AND ob.kod2=Pr.obor2
        INNERJOIN Obory_skup os
        ON os.kod=ob.kod
        WHERE Cl.funkce=1
)
GO
```

5.6.4 Publikace

Náhled na publikace vytvořené daným uživatelem. Poskytují přehled o základních i specifičtějších údajích týkající se daného díla.

```
CREATEVIEW poh_Publikace AS
(
    SELECT
        STUFF(
        (SELECTRTRIM(' '+prosp.prijmeni +' '+prosp.jmeno+',')
        FROM proces_spoluautori prosp
        WHERE prosp.publikace = p.ID
        FORXMLPATH (''))
        , 1, 1,'')AS'Autor',
P.nazev AS'publikace',YEAR(P.vydano)AS'rok', Vk.nazev AS'kategorie', Vd.nazev AS'typ', P.vydavatel, P.ISSN, P.ISBN, P.body
    FROM Publikace P
    INNERJOIN Vyz_druhy Vd
    ON P.druh=Vd.kod
    INNERJOIN Vyz_kategorie Vk
    ON Vk.ID=Vd.kategorie
)
GO
```

5.6.5 Přehled zkušeností

Roky pedagogické a jiné praxe, léta stáží, počet jazyků, jimiž hovoří, průměrná úroveň těchto jazyků. Úrovně jsou předdefinované A1, A2, B1, B2, C1, …

Přehled jazykových znalostí včetně ukázky:

```
CREATE VIEW poh_Jazyky AS

(

    SELECT J.nazev AS 'jazyk', Jur.uroven AS 'úroveò', Jur.nazev AS 'level', Zam.jmeno, Zam.prijmeni, Zam.osobni_c, Jzk.datum, Ex.nazev AS 'místo zkoušky'

    FROM Jaz_zkousky Jzk

    INNERJOIN Jaz_urovne Jur

    ON Jzk.uroven=Jur.ID_ur

    INNERJOIN Jazyky J

    ON J.ID_jaz=Jzk.ID_jaz

    INNERJOIN Zamestnanci Zam

    ON Zam.osobni_c=Jzk.ID_zamest

    LEFTJOIN Ext_organizace Ex

    ON Ex.ID_org=Jzk.misto

)

GO
```

Tabulka 3 – Ukázka výpisu z databáze – jazykové zkušenosti

jazyk	úroveň	level	jméno	příjmení	osobni_c	datum	místo zkoušky
Němčina	B2	pokročilý	Ondřej	Skoupil	123	16.8.2012	Jazyková instituce Brno
Němčina	B2	pokročilý	Petr	Doležal	333	16.8.2010	Jazyková instituce Brno
Angličtina	A2	začátečník	Pavel	Smutný	123	16.8.2012	Jazyková instituce Brno
Angličtina	B2	pokročilý	Ondřej	Skoupil	123	16.8.2010	Jazyková instituce Brno

Zdroj: vlastní zpracování

5.6.6 Statistické tabulky v rámci uživatele

Přehled o vývoji pracovníka v průběhu let, jež působí na fakultě. Počet vykonávaných funkcí, počet projektů, publikací, ZP, doktorandů, předmětů, jež učí a garantuje, počet odučených hodin, počet hodin strávených konzultacemi.

Ukázka části statistické tabulky. Je zde například vidět, že uživatel 222 je emeritním profesorem, vykoná každý týden deset hodin konzultací a je garantem pěti předmětů. Navíc každý týden odučí 63 hodin. Oproti tomu uživatel 333 vypracoval za poslední tři roky trojici projektů a vydal jednu publikaci.

Tabulka 4 – Ukázka výpisu z databáze – statistická tabulka v rámci uživatele

	osobni_c	Funkce	počet funkcí	konzultace	projekty	projekty >3 roky!	publikace	vedené předměty	odučeno hodin
1	123	doktorand	1	NULL	3	NULL	2	1	NULL
2	222	emeritní profesor	1	10	NULL	NULL	NULL	5	63
3	333	NULL	NULL	NULL	3	NULL	1	NULL	NULL
4	555	doktorand, asistent	2	NULL	2	NULL	NULL	NULL	NULL

Zdroj: vlastní zpracování

5.6.7 Statistické tabulky v rámci fakulty

Poskytují snadný pohled na data, jež jsou důležitá nejen pro plánování výuky. To vše za pomoci souhrnných částí odkazující se na různé části databáze.

Ukázka části statistické tabulky v rámci fakulty. Je zde například vidět, že pan Skácel sice neumí žádný cizí jazyk, ale na druhou stranu má za sebou 8 let pedagogické praxe, 6 let další praxe a absolvoval 1 rok stáže. Dále je možné tyto hodnoty porovnat s průměrem na dané fakultě a podle toho i směřovat další vývoj. Například panu Skácelovi by prospělo se naučit alespoň jeden jazyk na úrovni A2, aby nespadal pod průměr v jazykových znalostech.

Tabulka 5 – Ukázka výpisu z databáze – statistická tabulka v rámci fakulty

	osobní číslo	prijmeni	jazyků	roky ped. praxe	léta praxe	léta stáže
1	123	Skoupil	2	2	NULL	NULL
2	222	Skácel	0	8	6	1
3	333	Doležal	1	NULL	NULL	NULL
4	555	Miloševský	0	NULL	NULL	NULL

Zdroj: vlastní zpracování

5.7 Testování návrhu

Testování je důležité pro odhalení řady chyb a sjednání jejich nápravy. Zároveň musí být otestována základní funkčnost systému. S vývojem systému se očekává další testování a korekce chyb i na základě reportů od správců či jednotlivých uživatelů.

Pro účely testování je nutné vložit data. Data byla zvolena pokud možno tak, aby pokrývala všechny možnosti a varianty, jež by mohly nastat. Přičemž velká část je vložena i z důvodu samotné funkčnosti a bez těchto dat by systém nemohl plně fungovat.

5.7.1 Předvyplnění číselníků

Velkou část entit představují číselníky, u kterých většinou není vhodné, aby je vyplňoval uživatel. Z tohoto důvodu jsem vyplnil většinu podstatných údajů.

5.7.1.1 Vyhledávání v rámci lokality

V rámci základního dělení na kontinenty. Dále je možnost specifikovat dle jednotlivých států. Zkratky jednotlivých zemí jsou i primárními klíči. Podle zemí se přiděluje i předvolba pro telefonní čísla. Nejbližší specifikace lokality je možná dle českých měst. Aktuálně jsou v návrhu zahrnuty všechny státy včetně primárních telefonních předvoleb. Dále jsou v databázi uložena veškerá česká města včetně jejich částí a poštovního směrovacího čísla, jež jsou vedena v registru České pošty.

5.7.1.2 Tituly

Při vložení informací o absolvování je na výběr řada titulů, jak známé klasické, tak i čestné tituly. Tyto tituly jsou poté automaticky umístěny před nebo za jméno pracovníka.

5.7.1.3 Jazyky, Jazykové úrovně

Světové jazyky a všeobecně známé dělení dle jazykových úrovní (A1, A2, B1,…)

5.7.2 Imaginární data

Z důvodu testování funkčnosti je většina ostatních tabulek naplněna smyšlenými daty. Jakákoliv shoda je čistě náhodná. Data byla zvolena, tak aby pokrývala pokud možno všechny situace, jež by mohly nastat.

5.7.3 Postup testování

Po naplnění tabulek daty se přechází k vlastnímu testování. V prvním kroku je zkoumáno, zdali je možné vložit nepatřičná data a následně zamezit jejich vložení. Následuje zkoumání provázanosti dat a jejich správného zobrazení. V konečné fázi se zaměřuji na funkčnost jednotlivých prvků, jako jsou funkce, procedury a trigerry.

5.7.4 Výsledky testování

Po vložení dat byl systém otestován nejen v rámci funkčnosti, ale i korektnosti zobrazení v jednotlivých pohledech. Po mnohočetném testování byly na základě jednotlivých situací eliminovány různorodé chyby a bylo zamezeno jejich opakování. Další fází je zařazení do provozu a odstranění vad na základě feedbacku od jednotlivých uživatelů a správců systému.

5.7.5 Přínosy testování

Testování přispělo k razantnímu zlepšení kvality systému, přičemž byla také ověřena funkčnost návrhu a jeho použitelnost. Na základě výsledků lze interpretovat dílčí části systému a prokázat jejich použitelnost v praxi. Tím je tento návrh systému nyní připraven

na etapu dokončení. Po doplnění zbývajících částí, bude systém zcela funkční a připravený na ostrý provoz.

5.8 Základní prvky bezpečnosti navrhovaného systému

V této části jsou nastíněny základní prvky bezpečnosti. Nejsou probírány do detailů, jelikož nejsou plnohodnotným tématem této práce. Jedná se jen o nastínění cesty vývoje, kterou by se měl návrh dále ubírat.

5.8.1 Možná ohrožení

Z hlediska bezpečnosti ohrožuje databázi především poškození nebo smazání dat. Z tohoto důvodu by měla být vytvořena zrcadlená databáze vyskytující se na jiném serveru pro zastoupení v případě nefunkčnosti hlavního serveru. Dále je nutné data zálohovat, aby bylo možné v případě nutnosti tato data z daného časového úseku obnovit.

Přístup neautorizovaných osob může ohrozit důvěru v systém nepřátelskou činností. Proto je vhodné využít autorizaci osob a přidělení rolí jednotlivým uživatelům.

5.8.2 Přístup k databázi

Z hlediska bezpečnosti se uživatel přihlašuje pomocí svého přístupového jména a hesla. Neautorizovaní uživatelé nemají dále přístup. Vzhledem k roli autorizovaného uživatele jsou udělena práva pro přístup. Uživatel se tedy pohybuje jen v předem vytyčené oblasti s ošetřenými vstupy. V případě problému může kontaktovat správce, jenž má rozšířenou, ale stále omezenou působnost. Vzhledem k možnosti zneužití jsou zásahy správce evidovány. Pokud problém není vyřešen, přechází dále na administrátora, jenž není nijak omezen. Administrátor zodpovídá za stav systému a změnu dat po jeho zásahu.

Tabulka 6 – Přehled dostupných rolí

ROLE	vkládání dat	prohlížení dat	přístup k jiným účtům
uživatel	Jen za pomoci procedur	předdefinované pohledy	NE
správce	Rozšířená množina procedur	předdefinované pohledy	Omezeno
administrátor	neomezeno	neomezeno	ANO

Zdroj: vlastní zpracování

6 Přínos návrhu řešení

Byla vytvořena podpůrná databáze zaměstnanců na základě analyzovaných a zpracovaných požadavků popisu současné situace. Výstupem databáze jsou zde také utříděné pohledy jednotlivých agend a demonstrace výstupu dat formou tvorby strukturovaného životopisu jednotlivých pracovníků.

Výhodou tohoto systému je minimalizace problémů při práci s daty a nejednotnými záznamy. Veškerá data jsou dále archivována a umožňují pohled na data v rámci jejich historie. Po naprogramování aplikačního prostředí se předpokládá celkově ucelený a jednotný pohled na data, jenž poskytuje komfortní a pohodlnou práci s databází. Přístup k databázi tak navýší efektivitu dosavadních činností uživatele.

Nabídka jednotlivých funkcí je odvozena podle role uživatele a není nutné ani školit obsluhu. Ovládání by mělo být zcela intuitivní po přidání aplikační části, které v grafické formě s intuitivním ovládáním dosáhne uživatelsky přívětivé úrovně.

Jednou z největších výhod je praktická přehlednost jednotlivých agend za pomoci pohledů. Personalisté a také samotní zaměstnanci mohou kontrolovat množství času vynaloženého na danou agendu a podle toho patřičně pozměnit časový rozvrh. Vzniká tak praktická využitelnost pro plánování a usměrňování profesní způsobilosti jednotlivých pedagogů.

6.1 Omezení návrhu

V systému nejsou evidováni regulérní studenti a jejich známky. Chybí tím automatické evidování ročníků a aktuální počet studentů se musí každý rok aktualizovat ručně. Ovšem tento problém lze snadno vyřešit například importem dat.

6.2 Možnosti

Systém lze snadno rozšířit na různé fakulty i univerzity bez nutnosti měnit entity ani atributy. Pohledy je možné specifikovat pro každou fakultu či univerzitu zvlášť. V tomto

ohledu uživateli stačí přístup přes jeden systém a informace o něm budou zveřejněny pro veškeré univerzity, kde působí. Minimalizuje se tak množství data systém, byť jeden lze velmi snadno nabízet pro různé instituce.

Další kroky by určitě měly vést k vytvoření aplikačního prostředí pro příjemnější práci s databází. Aktuální podoba systému není vhodná pro široké spektrum uživatelů, jež nemají zkušenosti s SQL. Vytvořením přístupového portálu ve spolupráci s grafickým týmem se otevírají další možnosti nejen, jak zpříjemnit práci s databází.

7 Závěr

Celkově systém podporuje nejenom veškeré požadavky, jež byly v průběhu práce specifikovány, ale rozšiřuje množinu možností o další části, jež mohou usnadnit a zefektivnit dosavadní práci pedagogů a vědeckých pracovníků.

Cílem bylo úspěšné vytvoření databázového systému pro podporu personální agendy. Toho bylo docíleno vytvořením vhodné databázové struktury. Ta byla dále rozdělena do několika podoblastí. Tyto části byly dále specifikovány z hlediska jednotlivých entit a relací.

Databáze primárně obsahuje evidenci základních údajů jednotlivých vyučujících. Jednak eviduje základní údaje o předmětech z hlediska historických záznamů a podpory plánovaní výuky, jednak i přehled o letitě nabytých zkušenostech v rámci pedagogické a pracovní oblasti jednotlivých pedagogů.

Na data lze nahlížet za pomoci uživatelsky přívětivých pohledů. Tyto pohledy jsou rozděleny na uživatelské a pohledy pro správu lidských zdrojů.

Některé z těchto pohledů je nutné specifikovat. To je často zajištěno i za pomoci funkcí. Funkce, ale i procedury a triggery napomáhají plynulému a automatizovanému chodu systému. Zároveň zajišťují i bezpečnost systému, poněvadž každý uživatel má jen svou předdefinovanou množinu procedur, jež je mu přidělena podle jeho role.

Systém lze dále rozvíjet, a to nejen v rámci jedné univerzity. Je možné jej nabídnout i jiným vysokým či středním školám pro práci s lidskými zdroji.

8 Seznam použitých zdrojů

[1] American Economic Association. EconLit. *aeaweb.org* [online]. [cit. 2014-11-11]. Dostupné z: https://www.aeaweb.org/econlit/jelCodes.php

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

[3] ČESKO. Zákon č. 111 ze dne 22. dubna 1998 o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách). In: *Sbírka zákonů České republiky*. 1998, částka 39, s. 5388-5419. Dostupný také z: http://www.msmt.cz/vzdelavani/vysoke-skolstvi/zakon-c-111-1998-sb-o-vysokych-skolach-text-se-zapracovanymi. ISSN 1211-1244.

[4] DATE, C. J. *An Introduction to Database Systems.* Pearson/Addison-Wesley, 2004. 1005 s. ISBN 0-321-19784-4.

[5] HARDCASTLE, Elizabeth. *Business Information Systems.* Telluride: Ventus Publishing, 2008. 56 s. ISBN 978-87-7681-46.

[6] HOTEK, M. *Microsoft SQL server 2008: Krok za krokem.* 1. vyd. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6.

[7] CHURCHER, C. *Beginning Database Design.* 2. vyd. New York City: Apress, 2012. 252 s. ISBN 978-1-4302-4209-3.

[8] KOCH, M. *Datové a funkční modelování.* 1. vyd. Brno: CERM, 2004. 108 s. ISBN 80-214-2724-8.

[9] KŘÍŽ, J., DOSTÁL, P. *Databázové systémy.* 1. vyd. Brno: CERM, 2005. 111 s. ISBN 80-214-3064-8.

[10] LACKO, Luboslav. *Jak vyzrát na Microsoft SQL Server 2008.* Brno: Computer Press, 2009. 472 s. ISBN 978-80-251-2101-6.

[11] LACKO, Luboslav. *SQL Kapesní přehled.* Brno: CP Books, 2005. 96 s. ISBN 80-251-0788-4.

[12] MISTRY, Ross a Stacia MISNER. *Introducing Microsoft SQL Server 2008 R2*. Redmond: Microsoft Press, 2010. 263 s. LCCN: 2010925075.

[13] OPPEL, J. A. *Databáze bez předchozích znalostí*. Brno: Computer Press, 2006. 320s. ISBN 80-251-1199-7.

[14] OPPEL, J. A. *SQL bez předchozích znalostí.* 1. vyd. Brno: Computer Press, 2008. 240 s. ISBN 978-80-251-1707-1.

[15] STEPHENS, R, R. PLEW a A. D. JONES. *Naučte se SQL za 28 dní.* Brno: Computer Press, 2010. 728 s. ISBN 978-80-251-2700-1.

9 Seznam tabulek a obrázků

10 Seznam příloh

Příloha č. 1: zdrojový kód fyzického návrhu databáze

Příloha č. 2: zdrojový kód návrhu výpisu životopisu

11 Přílohy

11.1 Příloha č. 1 : zdrojový kód fyzického návrhu databáze

```
/*      POZNAMKY

Použité zkratky:
      Vyz - výzkum
      Ped - pedagogika
      Ext - externí
      Prac - pracovní
      Dot - dotazník
*/


------------------------ START ------------------------
PRINT 'Vytváøím databázi OndraS'
IF EXISTS (SELECT 1 FROM SYS.DATABASES WHERE NAME = 'OndraS')
DROP DATABASE OndraS
GO

CREATE DATABASE OndraS
GO


USE OndraS
GO

PRINT 'Vytváøím tabulky'
GO
--

------------------------ Ciselniky a pruvodni tabulky -------------------
-----

CREATE TABLE Dny
(
      ID_den SMALLINT,
      nazev VARCHAR (7) NOT NULL,
      PRIMARY KEY (ID_den)
)
GO

CREATE TABLE Funkce
(
      ID SMALLINT identity (1, 1),
      nazev VARCHAR (30) NOT NULL,
      body SMALLINT NOT NULL,
      PRIMARY KEY (ID),
)
GO
```

```
CREATE TABLE Tituly
(
      ID SMALLINT identity (1, 1),
      zkratka CHAR (10) NOT NULL,
      nazev VARCHAR(35) NOT NULL,
      pozice BIT,
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Jazyky
(
      ID_jaz INT identity (1, 1),
      nazev VARCHAR (20) NOT NULL,
      PRIMARY KEY (ID_jaz)
)
GO

CREATE TABLE Jaz_urovne
(
      ID_ur INT identity(1, 1),
      uroven VARCHAR(13) NOT NULL,
      nazev VARCHAR (18) NOT NULL,
      PRIMARY KEY (ID_ur)
)
GO

----------------------- GLOBALNI STRUKTURA -----------------------

CREATE TABLE Kontinenty
(
      ID INT identity (1, 1),
      nazev VARCHAR (19) NOT NULL,
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Zeme
(
      zkratka CHAR (2) NOT NULL,
      nazev VARCHAR (26) NOT NULL,
      kontinent INT NOT NULL DEFAULT (1),
      tel_predvolba VARCHAR(6) NOT NULL,
      PRIMARY KEY (zkratka),
      FOREIGN KEY (kontinent)
      REFERENCES Kontinenty(ID)
)
GO

CREATE TABLE Ext_organizace
(
      ID_org INT identity(1, 1),
      nazev VARCHAR (70) NOT NULL,
```

```
      email VARCHAR (50),
      www VARCHAR (50),
      popis TEXT,
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (ID_org)
)
GO

----------------------- ADRESY -----------------------

CREATE TABLE Mesta
(
      nazev VARCHAR (45) NOT NULL,
      PSC VARCHAR(10) NOT NULL,
      zeme CHAR(2) DEFAULT ('CZ') NOT NULL,
      PRIMARY KEY (PSC,zeme),
      FOREIGN KEY (zeme)
      REFERENCES Zeme(zkratka)
)
GO

CREATE TABLE Adresy
(
      ulice VARCHAR (30) NOT NULL,
      cp VARCHAR (12) NOT NULL,
      PSC VARCHAR(10) NOT NULL,
      zeme CHAR(2) NOT NULL DEFAULT ('CZ'),
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (PSC,cp),
      FOREIGN KEY (PSC,zeme)
      REFERENCES Mesta(PSC,zeme)
)
GO

CREATE TABLE Mista
(
      ID_org INT NOT NULL,
      cp VARCHAR(12) NOT NULL,
      PSC VARCHAR(10) NOT NULL,
      tel CHAR (9) NOT NULL,
      tel2 CHAR (9),
      zeme CHAR (2) DEFAULT ('CZ') NOT NULL,
      vlozeno DATE NOT NULL DEFAULT GETDATE(),
      PRIMARY KEY (ID_org,cp,PSC),
      FOREIGN KEY (PSC,cp)
      REFERENCES Adresy(PSC,cp),
      FOREIGN KEY (ID_org)
      REFERENCES Ext_organizace(ID_org),
      FOREIGN KEY (zeme)
      REFERENCES Zeme(zkratka)
)
GO
```

```
------------------------ INTERNÍ STRUKTURA ------------------------

CREATE TABLE Univerzity
(
      zk_u VARCHAR (10) NOT NULL,
      nazev VARCHAR (80) NOT NULL,
      PRIMARY KEY (zk_u)
)
GO

CREATE TABLE Fakulty
(
      zk_f VARCHAR (4) NOT NULL,
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      nazev_fakulty VARCHAR (51) NOT NULL,
      PRIMARY KEY (zk_f,zk_u),
      FOREIGN KEY (zk_u)
      REFERENCES Univerzity (zk_u)
)
GO

CREATE TABLE F_kontakty
(
      zk_f VARCHAR (4) NOT NULL,
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      vlozeno DATE NOT NULL DEFAULT GETDATE(),
      ulice VARCHAR (30) NOT NULL,
      cp CHAR(8) NOT NULL,
      PSC VARCHAR(10) NOT NULL,
      zeme CHAR(2) NOT NULL DEFAULT ('CZ'),
      tel VARCHAR (9) NOT NULL,
      PRIMARY KEY (zk_f,PSC),
      FOREIGN KEY (PSC,zeme)
      REFERENCES Mesta(PSC,zeme),
      FOREIGN KEY (zk_f,zk_u)
      REFERENCES Fakulty(zk_f,zk_u)
)
GO

CREATE TABLE Ustavy
(
      ID_ustav INT NOT NULL,
      fakulta VARCHAR (4) NOT NULL,
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      nazev VARCHAR (30)NOT NULL,
      PRIMARY KEY (ID_ustav),
      FOREIGN KEY (fakulta,zk_u)
      REFERENCES Fakulty(zk_f,zk_u)
)
GO

CREATE TABLE Kancelare
(
```

```
	zkratka CHAR (4) NOT NULL,
	klapka SMALLINT NOT NULL CHECK(klapka>0),
	pocet_mist SMALLINT NOT NULL CHECK(pocet_mist>0),
	ustav INT NOT NULL,
	PRIMARY KEY (ustav,zkratka),
	FOREIGN KEY (ustav)
	REFERENCES Ustavy(ID_ustav)
)
GO

----------------------- ZAMESTNANCI -----------------------

CREATE TABLE Zamestnanci
(
	uziv_jmeno VARCHAR(20) NOT NULL UNIQUE,
	jmeno VARCHAR (20) NOT NULL,
	prijmeni VARCHAR (20) NOT NULL,
	--rc VARCHAR(6) NOT NULL,
	osobni_c INT NOT NULL,
	c_dodavatele_SAP INT,
	ridicsky_p BIT,
	email VARCHAR (50) NOT NULL,
	FAX VARCHAR (9),
	vlozeno DATE NOT NULL DEFAULT GETDATE(),
	smazano BIT NOT NULL DEFAULT (0),
	smazano_d DATE,
	zmena DATE DEFAULT GETDATE(),
	PRIMARY KEY (osobni_c)
)
GO

CREATE TABLE Bydliste
(
	ID_zamest INT NOT NULL,
	ulice VARCHAR (50),
	cp VARCHAR (12) NOT NULL,
	PSC VARCHAR(10) NOT NULL,
	zeme CHAR(2) NOT NULL DEFAULT ('CZ'),
	poznamka VARCHAR (70),
	vlozeno DATE NOT NULL DEFAULT GETDATE(),
	smazano BIT NOT NULL DEFAULT (0),
	smazano_d DATE,
	PRIMARY KEY (ID_zamest,cp,PSC,zeme),
	FOREIGN KEY (ID_zamest)
	REFERENCES Zamestnanci(osobni_c),
	FOREIGN KEY (PSC,zeme)
	REFERENCES Mesta(PSC,zeme)
)
GO

CREATE TABLE Telefony
(
	zeme CHAR (2) DEFAULT ('CZ') NOT NULL,
	cislo CHAR (9) NOT NULL,
```

```
	ID_zamest INT NOT NULL,
	ucel VARCHAR (20),
	vlozeno DATE NOT NULL DEFAULT GETDATE(),
	smazano BIT NOT NULL DEFAULT (0),
	smazano_d DATE,
	PRIMARY KEY (zeme,cislo,ID_zamest),
	FOREIGN KEY (zeme)
	REFERENCES Zeme(zkratka),
	FOREIGN KEY (ID_zamest)
	REFERENCES Zamestnanci(osobni_c)
)
GO

CREATE TABLE Jmenovani -- vložení nadøízeným / personálním pracovníkem
(
	c_pomeru INT NOT NULL,
	ID_zamest INT NOT NULL,
	funkce SMALLINT NOT NULL,
	nadrizeny INT SPARSE NULL,
	zk_f VARCHAR (4) NOT NULL,
	zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
	ustav INT NOT NULL,
	kancelar CHAR (4),
	datum DATE NOT NULL DEFAULT GETDATE(),
	uvazek_od DATE NOT NULL DEFAULT GETDATE(),
	PRIMARY KEY (c_pomeru,funkce,zk_u),
	FOREIGN KEY (ID_zamest)
	REFERENCES Zamestnanci(osobni_c),
	FOREIGN KEY (ustav)
	REFERENCES Ustavy(ID_ustav),
	FOREIGN KEY (funkce)
	REFERENCES Funkce(ID),
	FOREIGN KEY (zk_f,zk_u)
	REFERENCES Fakulty(zk_f,zk_u),
	FOREIGN KEY (nadrizeny)
	REFERENCES Zamestnanci(osobni_c),
	FOREIGN KEY (ustav,kancelar)
	REFERENCES Kancelare(ustav,zkratka)
)
GO

CREATE TABLE Konzultace
(
	ID_zamest INT NOT NULL,
	mistnost CHAR (4) NOT NULL,
	ustav INT NOT NULL,
	den SMALLINT NOT NULL,
	od TIME NOT NULL,
	do TIME NOT NULL,
	popis VARCHAR (50),
	PRIMARY KEY(ID_zamest,den,od),
	FOREIGN KEY (ID_zamest)
	REFERENCES Zamestnanci(osobni_c),
	FOREIGN KEY (ustav,mistnost)
```

```
      REFERENCES Kancelare(ustav,zkratka)
)
GO

----------------------- Dotazniky -----------------------

CREATE TABLE Dotazniky
(
      ID_dotaz INT identity(1, 1),
      nazev varchar (30) NOT NULL UNIQUE,
      popis TEXT,
      ID_zamest INT NOT NULL,
      zahajeno BIT NOT NULL,
      odpovedi SMALLINT SPARSE NULL,
      od DATE NOT NULL DEFAULT GETDATE(),
      do DATE NOT NULL,
      PRIMARY KEY (ID_dotaz),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c)
)
GO

CREATE TABLE Dot_otazky
(
      ID_dotaz INT NOT NULL,
      ID_otazka SMALLINT identity(1, 1),
      otazka VARCHAR (50) NOT NULL,
      ano_ne BIT NOT NULL,
      PRIMARY KEY (ID_dotaz,ID_otazka),
      FOREIGN KEY (ID_dotaz)
      REFERENCES Dotazniky(ID_dotaz)
)
GO

CREATE TABLE Dot_odpovedi
(
      ID_dotaz INT NOT NULL,
      ID_otazka SMALLINT NOT NULL,
      ID_odpovedel INT NOT NULL,
      odpoved VARCHAR (50) SPARSE NULL,
      odpoved2 BIT SPARSE NULL,
      PRIMARY KEY (ID_dotaz,ID_otazka,ID_odpovedel),
      FOREIGN KEY (ID_dotaz,ID_otazka)
      REFERENCES Dot_otazky(ID_dotaz,ID_otazka),
      FOREIGN KEY (ID_odpovedel)
      REFERENCES Zamestnanci(osobni_c)
)
GO

----------------------- PROJEKTY -----------------------

CREATE TABLE Obory_skup
(
      kod CHAR (1) NOT NULL,
```

```
      nazev VARCHAR(19) NOT NULL,
      PRIMARY KEY (kod)
)
GO

CREATE TABLE Obory
(
      kod2 CHAR(1) NOT NULL,
      kod CHAR (1) NOT NULL,
      nazev VARCHAR(55) NOT NULL,
      PRIMARY KEY (kod,kod2),
      FOREIGN KEY (kod)
      REFERENCES Obory_skup(kod)
)
GO

CREATE TABLE Skupiny
(
      nazev varchar (50) NOT NULL,
      cil varchar (100),
      vytvorena DATE NOT NULL DEFAULT GETDATE(),
      PRIMARY KEY (nazev)
)
GO

CREATE TABLE Sku_funkce
(
      ID INT identity(1, 1),
      nazev varchar (10) NOT NULL UNIQUE,
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Clenstvi
(
      skupina varchar (50) NOT NULL,
      ID_zamest INT NOT NULL,
      pridan DATE NOT NULL DEFAULT GETDATE(),
      funkce INT NOT NULL DEFAULT (3),
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (skupina,ID_zamest),
      FOREIGN KEY (skupina)
      REFERENCES Skupiny(nazev),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (funkce)
      REFERENCES Sku_funkce(ID)
)
GO

CREATE TABLE Projekty
(
      ID_proj INT identity (1, 1),
```

```
      skupina varchar (50) NOT NULL,
      nazev VARCHAR (80) UNIQUE NOT NULL,
      mezinarodni BIT NOT NULL DEFAULT (0),
      obor CHAR (1) DEFAULT ('A') NOT NULL,
      obor2 CHAR (1) DEFAULT ('H') NOT NULL,
      cil TEXT NOT NULL,
      popis TEXT NOT NULL,
      zahajeni DATE NOT NULL DEFAULT GETDATE(),
      ukonceni DATE,
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (ID_proj),
      FOREIGN KEY (skupina)
      REFERENCES Skupiny(nazev),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2)
)
GO

------------------------ PUBLIKACE ------------------------

CREATE TABLE JEL_1
(
      zkratka CHAR (1) NOT NULL,
      nazev VARCHAR (82) NOT NULL,
      PRIMARY KEY (zkratka)
)
GO

CREATE TABLE JEL_2
(
      zkratka CHAR (1) NOT NULL,
      razeni SMALLINT NOT NULL,
      nazev VARCHAR (103) NOT NULL,
      PRIMARY KEY (zkratka,razeni),
      FOREIGN KEY (zkratka)
      REFERENCES JEL_1(zkratka)
)
GO

CREATE TABLE JEL_3
(
      zkratka CHAR (1) NOT NULL,
      razeni SMALLINT NOT NULL,
      razeni2 SMALLINT NOT NULL,
      nazev VARCHAR (150) NOT NULL,
      PRIMARY KEY (zkratka,razeni,razeni2),
      FOREIGN KEY (zkratka,razeni)
      REFERENCES JEL_2(zkratka,razeni)
)
GO

/*
      JEL_zkratka CHAR (1),
```

```
      JEL_razeni SMALLINT,
      JEL_razeni2 SMALLINT,

      FOREIGN KEY (JEL_zkratka,JEL_razeni,JEL_razeni2)
      REFERENCES JEL_3(zkratka,razeni,razeni2)
*/

CREATE TABLE Vyz_kategorie
(
      ID INT,
      nazev varchar (30) UNIQUE NOT NULL,
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Vyz_druhy
(
      kod CHAR(6) NOT NULL,
      nazev VARCHAR (118) UNIQUE NOT NULL,
      kategorie INT NOT NULL,
      PRIMARY KEY (kod),
      FOREIGN KEY (kategorie)
      REFERENCES Vyz_kategorie(ID)
)
GO

CREATE TABLE Publikace
(
      ID INT identity(1, 1),
      nazev varchar (80) NOT NULL,
      nazev_CZ VARCHAR (100),
      nazev_EN VARCHAR (100),
      abstrakt TEXT NOT NULL,
      abstrakt_CZ TEXT,
      abstrakt_EN TEXT,
      klicova_slova TEXT,
      stran SMALLINT,
      ID_proj INT,
      vydavatel VARCHAR (50) NOT NULL,
      druh CHAR (6) NOT NULL,
      vydano DATE NOT NULL DEFAULT GETDATE(),
      citace INT,
      ISSN VARCHAR (50),
      ISBN VARCHAR (50),
      odkaz VARCHAR (80),
      popis TEXT,
      body SMALLINT,
      PRIMARY KEY (ID),
      FOREIGN KEY (druh)
      REFERENCES Vyz_druhy(kod),
      FOREIGN KEY (ID_proj)
      REFERENCES Projekty(ID_proj)
)
GO
```

```
CREATE TABLE Pub_razeni
(
      ID_pub INT NOT NULL,
      obor CHAR (1) DEFAULT ('A') NOT NULL,
      obor2 CHAR (1) DEFAULT ('H') NOT NULL,
      JEL1 CHAR(1),
      JEL2 SMALLINT,
      JEL3 SMALLINT,
      PRIMARY KEY (ID_pub,obor2),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2),
      FOREIGN KEY (JEL1,JEL2,JEL3)
      REFERENCES JEL_3(zkratka,razeni,razeni2),
      FOREIGN KEY (ID_pub)
      REFERENCES Publikace(ID)
)
GO

CREATE TABLE Spoluautori
(
      ID_zamest INT NOT NULL,
      publikace INT NOT NULL,
      PRIMARY KEY (ID_zamest, publikace),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (publikace)
      REFERENCES Publikace(ID)
)
GO

------------------------ CERTIFIKATY, JAZYKOVE ZNALOSTI -----------------
-------

CREATE TABLE Certifikaty
(
      ID_cert INT identity (1, 1),
      nazev VARCHAR (20) NOT NULL,
      popis TEXT,
      obor CHAR (1) DEFAULT ('A') NOT NULL,
      obor2 CHAR (1) DEFAULT ('I') NOT NULL,
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (ID_cert),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2)
)
GO

CREATE TABLE Certifikace
(
      ID_zamest INT NOT NULL,
      ID_cert INT NOT NULL,
      datum DATE NOT NULL DEFAULT GETDATE(),
```

```
      organizace INT NOT NULL,
      PRIMARY KEY (ID_zamest,ID_cert,datum),
      FOREIGN KEY (ID_cert)
      REFERENCES Certifikaty(ID_cert),
      FOREIGN KEY (organizace)
      REFERENCES Ext_organizace(ID_org),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c)
)
GO

CREATE TABLE Jaz_zkousky
(
      ID_jaz INT NOT NULL,
      ID_zamest INT NOT NULL,
      uroven INT NOT NULL,
      misto INT NOT NULL,
      datum DATE NOT NULL,
      ID_cert INT NOT NULL,
      PRIMARY KEY (ID_jaz,ID_zamest,uroven),
      FOREIGN KEY (ID_cert)
      REFERENCES Certifikaty(ID_cert),
      FOREIGN KEY (ID_jaz)
      REFERENCES Jazyky(ID_jaz),
      FOREIGN KEY (uroven)
      REFERENCES Jaz_urovne(ID_ur),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (misto)
      REFERENCES Ext_organizace(ID_org)
)
GO

CREATE TABLE Ostatni                        -- pro jakekoliv dalsi polozky
jez by chtel zamestanec uvest ve sve slozce
(
      ID INT identity (1, 1),          -- napø. soutìže, rekordy,...
      ID_zamest INT NOT NULL,
      datum DATE NOT NULL,
      nazev VARCHAR (50) NOT NULL,
      popis TEXT,
      odkaz VARCHAR (50)
      PRIMARY KEY (ID)
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c)
)
GO


------------------------ ZKUSENOSTI ------------------------

CREATE TABLE Uni_kat
(
      ID INT identity (1, 1),
```

```
      nazev VARCHAR (150),
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Univerzitni_aktivity
(
      ID_zamest INT NOT NULL,
      od DATE NOT NULL DEFAULT GETDATE(),
      do DATE,
      misto VARCHAR (4) NOT NULL,
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      kategorie INT NOT NULL,
      nazev VARCHAR (50) NOT NULL,
      PRIMARY KEY (ID_zamest,od,misto,nazev),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (misto,zk_u)
      REFERENCES Fakulty (zk_f,zk_u),
      FOREIGN KEY (kategorie)
      REFERENCES Uni_kat (ID)
)
GO

CREATE TABLE Prac_pozice
(
      ID INT identity(1, 1),
      nazev VARCHAR(30) NOT NULL,
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Prac_zkus
(
      ID_zamest INT NOT NULL,
      od DATE NOT NULL,
      do DATE,
      misto INT NOT NULL,
      pozice INT NOT NULL,
      PRIMARY KEY (ID_zamest,od,misto,pozice),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (misto)
      REFERENCES Ext_organizace(ID_org),
      FOREIGN KEY (pozice)
      REFERENCES Prac_pozice(ID)
)
GO

CREATE TABLE Typy_studia
(
      ID SMALLINT identity (1, 1),
      nazev VARCHAR (33) NOT NULL,
      PRIMARY KEY (ID)
```

```
)
GO

CREATE TABLE Ped_zkus
(
      ID_zamest INT NOT NULL,
      od DATE NOT NULL,
      do DATE NOT NULL,
      misto INT DEFAULT (1) NOT NULL,
      obor CHAR (1) NOT NULL,
      obor2 CHAR (1) NOT NULL,
      funkce SMALLINT NOT NULL,
      typ_studia SMALLINT,
      napln VARCHAR(100),
      PRIMARY KEY (ID_zamest,od,misto,funkce),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (misto)
      REFERENCES Ext_organizace(ID_org),
      FOREIGN KEY (funkce)
      REFERENCES Funkce(ID),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2),
      FOREIGN KEY (typ_studia)
      REFERENCES Typy_studia(ID)
)
GO

CREATE TABLE Staze
(
      ID_zamest INT NOT NULL,
      od DATE NOT NULL,
      do DATE,
      misto INT NOT NULL,
      popis TEXT,
      PRIMARY KEY (ID_zamest,od,misto),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (misto)
      REFERENCES Ext_organizace(ID_org)
)
GO

------------------------ VZDELANI ------------------------

CREATE TABLE Absolvovani
(
      titul SMALLINT,
      ID_zamest INT NOT NULL,
      datum DATE NOT NULL,
      zobrazovat BIT NOT NULL DEFAULT (1),
      typ_studia SMALLINT NOT NULL,
      obor CHAR (1) NOT NULL,
      obor2 CHAR (1) NOT NULL,
```

```
      fakulta VARCHAR (4),
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      PRIMARY KEY (titul,ID_zamest,datum),
      FOREIGN KEY (titul)
      REFERENCES Tituly(ID),
      FOREIGN KEY (ID_zamest)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2),
      FOREIGN KEY (typ_studia)
      REFERENCES Typy_studia(ID),
      FOREIGN KEY (fakulta, zk_u)
      REFERENCES Fakulty(zk_f,zk_u)
)
GO

----------------------- Predmety -----------------------

CREATE TABLE Typ_cvik
(
      ID_typ SMALLINT identity(1, 1),
      nazev VARCHAR (30) NOT NULL,
      PRIMARY KEY (ID_typ)
)
GO

CREATE TABLE Pravidelnosti
(
      ID_pravidelnost SMALLINT identity(1, 1),
      nazev VARCHAR (20) NOT NULL,
      opak REAL NOT NULL,
      PRIMARY KEY (ID_pravidelnost)
)
GO

CREATE TABLE Predmety
(
      ID_garant INT NOT NULL,
      nazev VARCHAR(30) NOT NULL,
      zkratka VARCHAR(6) NOT NULL,
      otevren DATE  NOT NULL DEFAULT GETDATE(),
      obor CHAR (1) NOT NULL,
      obor2 CHAR (1) NOT NULL,
      vlozeno DATE NOT NULL DEFAULT GETDATE(),
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (zkratka),
      FOREIGN KEY (ID_garant)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2)
)
GO
```

```
CREATE TABLE Stud_obory
(
      kod CHAR(8) NOT NULL,
      nazev VARCHAR (150) NOT NULL,
      fakulta VARCHAR (4) NOT NULL DEFAULT ('FP'),
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      typ_studia SMALLINT NOT NULL,
      rocniku SMALLINT NOT NULL DEFAULT(3),
      PRIMARY KEY (kod),
      FOREIGN KEY (fakulta,zk_u)
      REFERENCES Fakulty (zk_f,zk_u),
      FOREIGN KEY (typ_studia)
      REFERENCES Typy_studia(ID)
)
GO

CREATE TABLE Rocniky
(
      rocnik SMALLINT NOT NULL,
      stud_obor CHAR(8) NOT NULL,
      studentu SMALLINT NOT NULL,
      PRIMARY KEY (rocnik,stud_obor),
      FOREIGN KEY (stud_obor)
      REFERENCES Stud_obory (kod)
)
GO

CREATE TABLE Vyuka
(
      stud_obor CHAR(8) NOT NULL,
      rocnik SMALLINT NOT NULL,
      predmet VARCHAR(6) NOT NULL,
      sem_letni BIT,
      PRIMARY KEY (predmet, rocnik, stud_obor, sem_letni),
      FOREIGN KEY (stud_obor)
      REFERENCES Stud_obory(kod),
      FOREIGN KEY (predmet)
      REFERENCES Predmety(zkratka)
)
GO

CREATE TABLE Ucebny
(
      ucebna VARCHAR (5),
      fakulta VARCHAR (4) NOT NULL DEFAULT ('FP'),
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      kapacita SMALLINT NOT NULL,
      PRIMARY KEY (ucebna,fakulta,zk_u),
      FOREIGN KEY (fakulta,zk_u)
      REFERENCES Fakulty (zk_f,zk_u)
)
GO

CREATE TABLE Hodiny
```

```
(
      ID_cvicici INT,
      predmet VARCHAR(6) NOT NULL,
      den SMALLINT NOT NULL,
      od TIME NOT NULL,
      doba SMALLINT NOT NULL DEFAULT (2),
      mistnost VARCHAR (5) NOT NULL,
      fakulta VARCHAR (4) NOT NULL DEFAULT ('FP'),
      zk_u VARCHAR (10) NOT NULL DEFAULT ('VUT'),
      typ_cvika SMALLINT NOT NULL DEFAULT (2),
      pravidelnost SMALLINT,
      semestr BIT NOT NULL,
      smazano BIT NOT NULL DEFAULT (0),
      smazano_d DATE,
      PRIMARY KEY (ID_cvicici,den,od,semestr),
      FOREIGN KEY (den)
      REFERENCES Dny(ID_den),
      FOREIGN KEY (typ_cvika)
      REFERENCES Typ_cvik(ID_typ),
      FOREIGN KEY (ID_cvicici)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (pravidelnost)
      REFERENCES Pravidelnosti(ID_pravidelnost),
      FOREIGN KEY (mistnost,fakulta,zk_u)
      REFERENCES Ucebny (ucebna,fakulta,zk_u),
      FOREIGN KEY (predmet)
      REFERENCES Predmety(zkratka)
)
GO

------------------------ ZAVERECNE PRACE ------------------------

CREATE TABLE Typy_praci
(
      ID SMALLINT identity (1, 1),
      nazev VARCHAR (30) NOT NULL,
      obtiznost SMALLINT NOT NULL,
      PRIMARY KEY (ID)
)
GO

CREATE TABLE Zav_prace
(
      typ_prace SMALLINT NOT NULL,
      rok DATE NOT NULL DEFAULT GETDATE(),
      predmet VARCHAR (8),
      obor CHAR (1) NOT NULL,
      obor2 CHAR (1) NOT NULL,
      ID_vedouci INT NOT  NULL,
      ID_student INT NOT NULL,
      nazev varchar (50) NOT NULL,
      obhajeno BIT,
      hodnoceni_opo CHAR(1),
      hodnoceni_ved CHAR(1),
```

```
      hodnoceni_zav CHAR(1),
      PRIMARY KEY (rok,ID_student,obor,obor2),
      FOREIGN KEY (ID_vedouci)
      REFERENCES Zamestnanci(osobni_c),
      FOREIGN KEY (obor,obor2)
      REFERENCES Obory(kod,kod2),
      FOREIGN KEY (typ_prace)
      REFERENCES Typy_praci(ID)
)
GO
-----------------------------------------------------------------

PRINT ('Tabulky vytvoøeny')

USE MASTER
GO
```

11.2 Příloha č. 2 : zdrojový kód návrhu výpisu životopisu

```
USE OndraS
GO

------------------------------------------------------------------------
-------------------------------------------------     Osobní informace
CREATE PROCEDURE MY_ziv
@ID_in INT
AS
BEGIN
      IF (SELECT Zamestnanci.jmeno FROM Zamestnanci WHERE
osobni_c=@ID_in) is NULL
      PRINT ('Uživatel nenalezen')
      ELSE
      BEGIN
            DECLARE @jmeno VARCHAR (20) = (SELECT Zamestnanci.jmeno FROM
Zamestnanci WHERE osobni_c=@ID_in)
            DECLARE @prijmeni VARCHAR (20) = (SELECT Zamestnanci.prijmeni
FROM Zamestnanci WHERE osobni_c=@ID_in)
            DECLARE @titulp VARCHAR (100) = (SELECT [Titul pred] pred
FROM poh_Zamestnanci WHERE osobni_c=@ID_in)
            DECLARE @titulz VARCHAR (100) = (SELECT [Titul za] FROM
poh_Zamestnanci WHERE osobni_c=@ID_in)
            PRINT ('1. Osobní informace')
            PRINT ('   osobní èíslo: ')
                  +' '+convert(varchar(10),@ID_in)
            PRINT ('   Jmenovka: ')
                  +' '+COALESCE(@titulp,'')
                  +' '+@jmeno
                  +' '+@prijmeni
                  +' '+COALESCE(@titulz,'')

            IF (SELECT TOP 1 PSC FROM Bydliste WHERE ID_zamest=@ID_in) is
NOT NULL
            BEGIN
                  PRINT ('   Bydlištì: ')
                  DECLARE kurzor_bydliste
                        cursor for SELECT
                              B.cp, B.ulice, M.nazev FROM Bydliste B
                              INNER JOIN Mesta M
                              ON M.PSC=B.PSC
                              WHERE B.ID_zamest=@ID_in
                        declare @cp VARCHAR (12)
                        declare @ulice VARCHAR (50)
                        declare @nazev VARCHAR (45)
                        open kurzor_bydliste
                        fetch kurzor_bydliste
                        into @cp, @ulice, @nazev
                        while @@FETCH_STATUS = 0
                        begin
                              PRINT +'      '+@ulice
                                    +' '+@cp
                                    +' '+@nazev
```

```
                              fetch kurzor_bydliste
                              into @cp, @ulice, @nazev
                        end
                        close kurzor_bydliste
                  deallocate kurzor_bydliste
            END

            DECLARE @email VARCHAR (50) = (SELECT Zamestnanci.email FROM
Zamestnanci WHERE osobni_c=@ID_in)
            IF @email is NOT NULL
            BEGIN
                  PRINT('   Email: ')
                        +' '+@email
            END

            IF (SELECT TOP 1 ID_zamest FROM Telefony WHERE
ID_zamest=@ID_in) is NOT NULL
            BEGIN
                  DECLARE kurzor_telefony
                  cursor for  SELECT TOP 5 Zeme.tel_predvolba+'
'+SUBSTRING(Tel.cislo,1,3)+' '+SUBSTRING(Tel.cislo,4,3)+'
'+SUBSTRING(Tel.cislo,7,3) AS 'telefonní èíslo', Tel.ucel AS 'úèel'
                              FROM Telefony Tel
                              INNER JOIN Zamestnanci Zam
                              ON Zam.osobni_c=Tel.ID_zamest
                              INNER JOIN Zeme
                              ON Zeme.zkratka=Tel.zeme
                              WHERE Zam.osobni_c=@ID_in
                        declare @tel VARCHAR (16)
                        declare @ucel VARCHAR (20)
                        declare @poradi INT=1
                        open kurzor_telefony
                        fetch kurzor_telefony
                        into @tel, @ucel
                        while @@FETCH_STATUS = 0
                        begin
                              If @poradi=1
                                    Begin
                                          PRINT +'   Telefon:  '+@tel
                                                +' '+COALESCE(@ucel,'')
                                          SET @poradi=@poradi+1
                                          fetch kurzor_telefony
                                          into @tel, @ucel
                                    end
                                    else
                                    begin
                                                PRINT +'
Telefon'+CONVERT(varchar(2),@poradi)+': '+@tel
                                                      +'
'+COALESCE(@ucel,'')
                                                SET @poradi=@poradi+1
                                                fetch kurzor_telefony
                                                into @tel, @ucel
                                    end
```

```
                        end
                        close kurzor_telefony
                  deallocate kurzor_telefony
            END

            PRINT('')
      END

-------------------------------------------------------------------------
-------------------------------------------------     Pedagogická
èinnost
      IF (SELECT TOP 1 ID_zamest FROM Ped_zkus WHERE ID_zamest=@ID_in) is
NOT NULL
      BEGIN
      PRINT('2. pedagogická èinnost')
      PRINT +'   '+'obor         '+'od   '+'do   '+'Univerzita
'+'funkce                        '+'typ'
            DECLARE kurzor_ped_zkus
                  cursor for  SELECT ob.nazev AS 'obor', YEAR(pz.od) AS
'od', YEAR(pz.do) AS 'do', eo.nazev AS 'misto',
                                    fu.nazev AS 'funkce' ,ts.nazev AS
'typ studia', pz.napln
                              FROM Ped_zkus pz
                              LEFT JOIN Ext_organizace eo
                              ON eo.ID_org=pz.misto
                              INNER JOIN Funkce fu
                              ON fu.ID=pz.funkce
                              LEFT JOIN Obory ob
                              ON ob.kod=pz.obor AND ob.kod2=pz.obor2
                              INNER JOIN Typy_studia ts
                              ON ts.ID=pz.typ_studia
                              WHERE pz.ID_zamest=@ID_in
                              ORDER BY ob.nazev,pz.od
                        declare @obor VARCHAR (55)
                        declare @od CHAR (4)
                        declare @do CHAR (4)
                        declare @eo CHAR (70)
                        declare @funkce CHAR (30)
                        declare @studium CHAR (33)
                        declare @napln VARCHAR(100)
                        open kurzor_ped_zkus
                        fetch kurzor_ped_zkus
                        into @obor, @od, @do, @eo, @funkce, @studium,
@napln
                        while @@FETCH_STATUS = 0
                        begin
                              PRINT +'   '+@obor
                                    +' '+@od
                                    +' '+@do
                                    +' '+@eo
                                    +' '+@funkce
                                    +' '+COALESCE(@studium,'')
                                    +' '+COALESCE(@napln,'')
                              fetch kurzor_ped_zkus
```

```
                              into @obor, @od, @do, @eo, @funkce,
@studium, @napln
                        end
                        close kurzor_ped_zkus
                  deallocate kurzor_ped_zkus
      PRINT('')
      END

-------------------------------------------------------------------------
-------------------------------------------------     Pracovní
zkušenosti
      IF (SELECT TOP 1 ID_zamest FROM Prac_zkus WHERE ID_zamest=@ID_in)
is NOT NULL
      BEGIN
            PRINT('3. Pracovní zkušenosti')
            PRINT +'      '+'od   '+'do   '+'místo
'+'pozice'
            DECLARE kurzor_prac_zkus
                  cursor for  SELECT YEAR(pz.od) AS 'od', YEAR(pz.do) AS
'do', eo.nazev AS 'misto',pp.nazev AS 'pozice' FROM Prac_zkus pz
                              LEFT JOIN Ext_organizace eo
                              ON eo.ID_org=pz.misto
                              INNER JOIN Prac_pozice pp
                              ON pp.ID=pz.pozice
                              WHERE pz.ID_zamest=@id_in
                              ORDER BY pz.od
                        declare @od2 CHAR (4)
                        declare @do2 CHAR (4)
                        declare @eo2 CHAR (70)
                        declare @pozice CHAR (30)
                        open kurzor_prac_zkus
                        fetch kurzor_prac_zkus
                        into @od2, @do2, @eo2, @pozice
                        while @@FETCH_STATUS = 0
                        begin
                              PRINT +'      '+@od2
                                    +' '+COALESCE(@do2,'****')
                                    +' '+@eo2
                                    +' '+@pozice
                              fetch kurzor_prac_zkus
                              into @od2, @do2, @eo2, @pozice
                        end
                        close kurzor_prac_zkus
                  deallocate kurzor_prac_zkus
      PRINT('')
      END
END
GO


USE master
GO
```