VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH DATABÁZE SQL PRO STOMATOLOGICKOU KLINIKU

PROPOSAL OF SQL DATABASE FOR A DENTAL CLINIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL JANATA
AUTHOR

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

BRNO 2010

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2009/2010
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Janata Michal

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 SQL pro stomatologickou kliniku

v anglickém jazyce:

Proposal of SQL Database for a Dental Clinic

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

DOSEDĚL, T. Počítačová bezpečnost a ochrana dat. 1. vydání. Brno: Computer Press, 2004. 190 s. ISBN 80-251-0106-1
HOTEK, M. Microsoft SQL Server 2008: krok za krokem. 1.vydání. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6
KOTLER, P. Marketing management. 1. vydání. Praha: Grada, 2007. 788 s. ISBN 978-80-247-1359-5
PUŽMANOVÁ, R. Moderní komunikační sítě od A do Z. 2. aktualizované vydání. Brno: Computer Press, 2006. 430 s. ISBN 80-251-1278-0
ŘEPA, V. Podnikové procesy. 2.rozšířené vydání. Praha: Grada, 2007. 281 s. ISBN 978-80-247-2252-8
VOŘÍŠEK, J. Principy a modely řízení podnikové informatiky. 1. vydání. Praha: Oeconomica, 2008. 446 s. ISBN 978-80-245-1440-6

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

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2009/2010.

L.S.

Ing. Jiří Kříž, Ph.D. Ředitel ústavu	doc. RNDr. Anna Putnová, Ph.D., MBA

V Brně, dne 22.04.2010

Brno University of Technology
Faculty of Business and Management

Academic year: 2009/2010
Institute of Informatics

BACHELOR'S THESIS ASSIGNMENT

Janata Michal

Managerial Informatics (6209R021)

Pursuant to Act. No. 111/1998 Coll., on Higher Education Institutions, and in accordance with the Rules for Studies and Examinations of the Brno University of Technology an Dean's Directive on Realization of Bachelor and Master Degree Programs, the director of the Institute of is submitting you a bachelor's thesis of the following title:

Proposal of SQL Database for a Dental Clinic

In the Czech language:

Návrh databáze SQL pro stomatologickou kliniku

Instruction:

Literature / Sources:

DOSEDĚL, T. Počítačová bezpečnost a ochrana dat. 1. vydání. Brno: Computer Press, 2004. 190 s. ISBN 80-251-0106-1
HOTEK, M. Microsoft SQL Server 2008: krok za krokem. 1.vydání. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6
KOTLER, P. Marketing management. 1. vydání. Praha: Grada, 2007. 788 s. ISBN 978-80-247-1359-5
PUŽMANOVÁ, R. Moderní komunikační sítě od A do Z. 2. aktualizované vydání. Brno: Computer Press, 2006. 430 s. ISBN 80-251-1278-0
ŘEPA, V. Podnikové procesy. 2.rozšířené vydání. Praha: Grada, 2007. 281 s. ISBN 978-80-247-2252-8
VOŘÍŠEK, J. Principy a modely řízení podnikové informatiky. 1. vydání. Praha: Oeconomica, 2008. 446 s. ISBN 978-80-245-1440-6

The supervisor of bachelor's thesis: Ing. Jiří Kříž, Ph.D.

Submission deadline bachelor's thesis is given by the Schedule of the Academic year 2009/2010.

L.S.

Ing. Jiří Kříž, Ph.D. Director of the Institute	doc. RNDr. Anna Putnová, Ph.D., MBA Dean of the Faculty

Brno, 22.04.2010

Abstrakt

Tato bakalářská práce se zabývá návrhem databáze pro stomatologickou kliniku, která by měla zjednodušit a zefektivnit pracovní podmínky na klinikách. V první části jsou uvedeny teoretické základy problematiky. Další kapitola je zaměřena na analýzu současné situace a upozorňuje na důležité problémy. Na závěr je předložen vlastní návrh řešení a shrnutí tohoto návrhu.

Klíčová slova

SQL, logický návrh databáze, data, informační systém, databáze, proces, procedura

Abstract

This bachelor's thesis deals with a proposal of database for a dental clinic that should simplify and streamline working conditions. In the first part are given theoretical foundations of the problem. The next chapter is focused on analyzing the current situation and highlights potential important problems. Finally is presented my own draft resolution and a summary of the proposal.

Key words

SQL, logical database design, information system, database, process, procedure

Bibliografická citace práce

JANATA, M. *Návrh databáze SQL pro stomatologickou kliniku.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2010. 78 s. Vedoucí bakalářské práce Ing. Jiří Kříž, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená diplomová 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 31. května 2010

……

Poděkování

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

Obsah

Úvod

I když se v současné době používá výpočetní technika a informační systémy téměř ve všech odvětvích a činnostech, najde se mezi stomatology mnoho lékařů, kteří i přesto, že osobní počítač vlastní, využívají výpočetní techniku pouze pro získávání různých nových informací nebo například pro styk s pojišťovnami. V zavedení informačních systémů do stomatologických klinik brání nejen finanční náročnost a poměrně malý výběr z nabízených aplikací pro stomatology, ale také určitá rutina.

Převážně starší lékaři mají svůj zaběhlý systém, který je podle nich nejlepší a vše nové je pro ně zbytečné. Ovšem nejsou to pouze lékaři z vyšší věkové kategorie, ale také mladí lékaři, kteří se buď od těch starších naučili jejich zavedený a léty prověřený způsob záznamu informací nebo jen mají určitý odpor k technice.

Ve své bakalářské práci se zaměřuji na návrh databáze, která bude jednoduchá, přehledná a to i přesto, že bude obsahovat všechny požadované funkce. Vymezení problému a určení základních cílů, kterých má být dosaženo, je obsahem první části této práce.

Druhá část je zaměřena na teoretická východiska potřebná k vyřešení tohoto problému. Nejedná se o vyčerpávající informace, ale o stručný přehled o programovacím jazyce SQL, databázích a postupu při jejich efektivní tvorbě.

Analýza problému a současné situace je název další části, v níž je rozebráno, jak práce na některých stomatologických klinikách a ordinacích stále probíhá. Jsou zde popsány problémy, s kterými se lékaři a sestry často setkávají, a stručný popis řešení těchto problémů. Informace z této kapitoly jsou výchozí pro samostatný návrh databáze. V závěru kapitoly se ještě nachází analýza některých programů pro stomatology.

Poslední a nejdůležitější část bakalářské práce je vlastní návrh řešení. Je zde popsáno vytvoření logického a fyzického návrhu včetně vztahů mezi jednotlivými tabulkami. Také jsou zde uvedeny příklady skriptů a stručné zhodnocení přínosů bakalářské práce.

1 Vymezení problému a cíle práce

1.1 Vymezení problému

Mnoho stomatologických ordinací stále využívá papírové kartotéky k evidenci údajů a záznamů o pacientovi a mnoho lékařů stále vyplňuje klasické papírové žádanky o odborná a rentgenová vyšetření. Počítačový trh sice nabízí několik aplikací pro stomatology, ale tyto programy jsou poměrně drahé a jsou tak obsáhlé, že naučit se vše, co nabízí, zabere několik dnů či týdnů.

Z toho vznikl požadavek vytvořit informační systém, jehož hlavním úkolem nebude překonat dlouho vyvíjené profesionální aplikace za desetitisíce korun, ale umožnit i méně náročným zubním lékařům využívat aplikaci, která bude v podstatě elektronickým zpravováním papírové kartotéky a která bude jednoduchá, propojí jednotlivé části databáze, zefektivní a zrychlí práci s daty a podstatně sníží náklady za tzv. „papírování“.

1.2 Cíle práce

Hlavním cílem této bakalářské práce je navrhnout databázi pro stomatologickou kliniku.

V databázi bude evidence jednotlivých lékařů a jejich pacientů, u kterých bude možné ukládat záznamy o vyšetření nebo zákroku v podobě zubního kříže a dekursu (lékařského zápisu). Do databáze bude možné ukládat rentgenové snímky v elektronické podobě a bude z ní možné tisknout sestavy, které budou sloužit jako výkaz pro zdravotní pojišťovny. Navržená databáze také bude vstupem pro případnou internetovou stránku, na které by mohly být zobrazeny volné termíny lékaře.

V bakalářské práci budou také popsány základní kroky a požadavky na návrh databáze a základní informace o jazyce SQL a jeho vlastnostech.

2 Teoretická východiska

Cílem této kapitoly je čtenáři popsat stručný přehled teorie z oblasti databází a jazyka SQL. Na těchto poznatcích se bude dále v bakalářské práci stavět.

2.1 Základní pojmy a definice

2.1.1 Informace

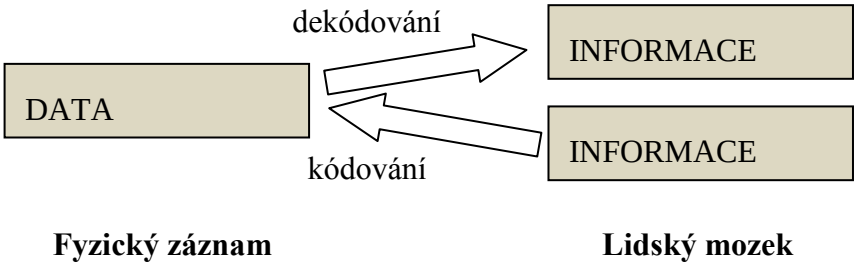
Pojem informace má mnoho významů z hlediska jednotlivých oborů, ve kterých se používá. Jedna z obecně platných definic říká, že informace je nějaká zpráva nebo údaj, který slouží ke snižování nejistoty.

„Informaci můžeme chápat jako zprávu, vjem, který 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 přijímací subjekty nějaký význam.“[1]

2.1.2 Data

Data jsou jakákoli vyjádření nějaké skutečnosti, schopná přenosu, interpretace či zpracování. *„Záznamem informace na vhodné médium (hovoříme o procesu kódování informace, tedy přesně definovaném postupu jak informaci zaznamenat) se informace stávají data, která opět přečtením (dekódováním) přejdou zpět na informaci pro daného příjemce.“*[2]

Obrázek 1 - Vztah informací a dat



Zdroj: [6] s.5

[1] KOCH, M. Datové a funkční modelování. 2004. s. 4.

[2] KOCH, M. Datové a funkční modelování. 2004. s. 5.

2.2 Databáze

Databáze je sjednocená sbírka logicky uspořádaných dat tak, aby se do nich nová data snadno ukládala, byla přehledná a aby se z nich snadno získávaly informace. Pojmem databáze je často myšlen celý databázový systém, který se skládá z více částí. Tak je tomu i v této práci.

2.2.1 Části databázového systému

„Obecně každý databázový systém se skládá z těchto částí: systému řízení báze dat (SŘBD), což je program, který organizuje a udržuje nashromážděné informace, z databázové aplikace, programu, který umožňuje vybírat, prohlížet a aktualizovat informace uložené prostřednictvím SŘBD a z databáze, čili bází dat (tj. uloženými daty).“[3]

SŘBD může být označován také jako DBMS z anglického Database Management System.

2.2.2 Datové modely

Datový model určuje strukturu dat v databázi a způsoby přístupu k těmto datům. Tyto modely prošly od 60. let 20. století do současnosti řadou změn.

2.2.2.1 Souborově orientovaný model

Souborově orientovaný systém je v podstatě elektronické zpracování ruční kartotéky vytvořené tak, aby bylo možné efektivněji zpracovávat data. Jednotlivé složky systému však pracují odděleně a nevytvářejí žádnou vazbu. V těchto systémech je velice obtížné vyhledávat, data nejsou konzistentní a je zde velká duplicita dat, což značně zvyšuje náklady na tento model. [1]

2.2.2.2 Hierarchický model

Tento model má klasickou stromovou strukturu. Mezi záznamy vzniká pomocí ukazatelů vztah rodič-potomek. Díky jednoduché struktuře byly tyto databáze snadno

[3] KŘÍŽ, J. *Databázové systémy.* 2005. s. 3.

pochopitelné a velice rychlé. Například systém IMS (Information Management Systém) společnosti IBM založený na hierarchickém modelu se pro aplikace, které mají za úkol zpracovat velká množství dat (např. bankovní transakce), používá dodnes. [3]

2.2.2.3 Síťový model

Síťový model je založen na stejném principu jako model hierarchický. Hlavní výhodou však je, že segmenty databáze můžou ukazovat na jiné segmenty v různých směrech. [6]

Příkladem takové databáze je například IDMS od společnosti Cullinet nebo Total od firmy Cincom.

2.2.2.4 Relační datový model

Relační model dat navrhl E. F. Codd v roce 1970. Silnou stránkou tohoto modelu je jednoduchá a logická struktura – všechna data jsou reprezentována pomocí relací, které jsou fyzicky vyjádřeny tabulkami.

2.2.2.5 Objektově datový model

„Nejnovějším datovým modelem je objektový datový model. Objektové datové modely jsou vystavěny na základním prvku – objektu (odpovídá přibližně pojmu věta), kde tento objekt má kromě svých atributů i definované metody, které určují chování objektu.“[4]

2.2.3 Relační datový model

Relační model dat obsahuje pět základních složek:

Relace – tabulka se sloupci a řádky.

Atribut – jeden sloupec relace.

Entita – jeden řádek tabulky.

Doména – množina možných hodnot pro atributy.

Relační databáze – kolekce normalizovaných tabulek.

[4] KOCH, M. Datové a funkční modelování. 2004. s. 23.

Relace se používají k uložení informací o objektech reprezentovaných v databázi. Jednotlivé pojmenované sloupce (atributy) relace určují, co budeme o objektech do databáze ukládat, entity jsou pak jednotlivé objekty. [1]

2.2.3.1 Pravidla relační databáze

Pokud má být databáze považována za relační musí podle Codda splňovat všechna tato pravidla [3]:

1. **Pravidlo informace** – všechny informace se reprezentují hodnotami v tabulkách.
2. **Pravidlo zaručeného přístupu** – každý údaj musí být přístupný pomocí názvu tabulky, názvu sloupce a hodnoty primárního klíče.
3. **Systematické ošetření prázdných hodnot** – podpora hodnoty NULL.
4. **Dynamický online katalog** – vyžaduje používání metadat.
5. **Pravidlo komplexního datového podjazyku** – musí existovat jazyk, který bude podporovat všechny funkce databáze.
6. **Pravidlo aktualizace pohledu** – systémová aktualizace pohledů (virtuálních tabulek).
7. **Vysokoúrovňové vkládání, aktualizace, odstranění** – vyžaduje, aby se řádky zpracovávaly jako množiny operací insert, delete a update.
8. **Fyzická datová nezávislost**
9. **Logická datová nezávislost** – 8 a 9 izolují uživatele nebo aplikaci od nízkoúrovňové implementace databáze.
10. **Nezávislost integrity** – jazyk a ne aplikační program by měl podporovat omezení integrity.
11. **Distribuční nezávislost** – jazyk musí být schopen pracovat s daty umístěnými na jiném počítačovém systému.
12. **Pravidlo nenarušení** – zabraňuje narušení relační struktury a integrity databáze.

2.2.3.2 Klíče relace

Pokud zajistíme, aby každý záznam (entita) v databázi byl jedinečný (žádná duplicita), pak lze ke každému záznamu přiřadit klíč, který ho jednoznačně identifikuje.

Kandidátní klíč – jeden nebo více (minimální počet) sloupců, které jednoznačně identifikují záznam. Kandidátní klíč musí být jednoznačný a neredukovatelný.

Primární klíč – jeden vybraný klíč z kandidátních klíčů.

Cizí klíč – klíč v jedné relaci, který je shodný s primárním klíčem v jiné relaci (ukazuje na něj).

2.2.3.3 Relační integrita

Entitní integrita – vztahuje se k primárním klíčům tabulek (relací). Určuje, že hodnota primárního klíče nesmí být prázdná (NULL).

Referenční integrita – zajišťuje, aby bylo možné vytvořit záznam pouze s takovým cizím klíčem, ke kterému existuje shodný primární klíč v jiné relaci, jinak musí být hodnota cizího klíče NULL.

Omezení domén – omezuje data, která lze do daného sloupce zapsat.

2.2.4 Normalizace

Normalizace je technika, která se používá pro vytvoření tabulek tak, aby se minimalizovala nadbytečnost dat.

1. normální forma

První normální forma je splněna, pokud každá buňka tabulky obsahuje právě jednu hodnotu. To znamená, že hodnota nesmí být složená ani vícehodnotová. Je možné se rozhodnout pro složenou hodnotu, pokud nepotřebujeme její jednotlivé části (např. adresa) a i tak zůstane 1. normální forma splněna. [1]

2. 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.

Druhá normální forma se týká jen tabulek se složenými primárními klíči, tedy tabulek, jejichž primární klíč tvoří dva nebo více sloupců.“[5]

3. normální forma – tranzitivní závislost (Boyce-Coddova normální forma)

Za třetí normální formu považujeme, pokud je relace v 1. i 2. normální formě a zároveň nejsou její atributy na sobě vzájemně závislé, kromě primárního klíče.

Boyce – Coddova normální forma je pouze variací třetí normální formy a platí, je-li relace v Boyce – Coddově normální formě je i ve třetí normální formě. Tato forma se zabývá složenými kandidátními klíči, když se v některých částech překrývají. [6]

4. normální forma

„Relace je čtvrté normální formě, pokud je v Boyce – Coddově normální formě, a navíc všechny vícehodnotové závislosti jsou zároveň funkčními závislostmi z kandidátních klíčů (v jedné relaci se nesmí spojovat nezávislé opakované skupiny).“[6]

5. normální forma

„Týká se případu spojené závislosti, která vyjadřuje cyklické omezení, pokud je relace 1 spojena s relací 2, relace 2 je spojena s relací 3 a relace 3 je spojena zpětně s relací 1, pak všechny tři entity musí být součástí stejného vektoru hodnot.“[7]

---

[5] CONOLLY, T. Mistrovství – databáze: Profesionální průvodce tvorbou efektivních databází. 2009. s. 192.

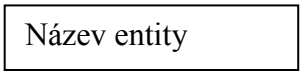
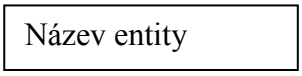
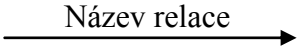
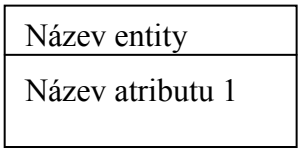
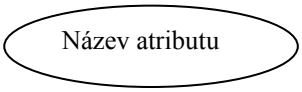
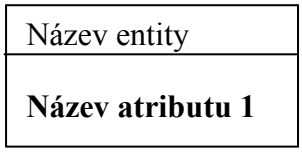
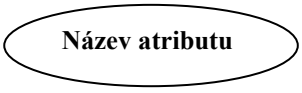
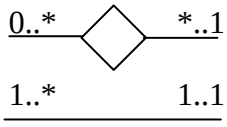
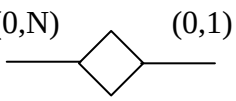
[6] KOCH, M. Datové a funkční modelování. 2004. s. 63.

[7] KOCH, M. Datové a funkční modelování. 2004. s. 64.

2.2.5 E-R model

„Entitně-relační modelování návrhu databáze odpovídá přístupu k návrhu metodou shora dolů. ER modelování začíná určením důležitých dat (nazývaných entity) a relací mezi daty, které je třeba v modelu reprezentovat. Pak se postupuje k podrobnostem jako například informacím, které je třeba o entitách a vztazích uchovávat (nazývaných atributy) a omezením platným pro entity, vztahy a atributy.“[8]

Tabulka 1 - Grafické znázornění základních prvků ER modelu

Prvek	1. Způsob	2. Způsob
Entita		
Binární relace		
Relace vyššího stupně		
Atribut		
Primární klíč		
Multiplicita		

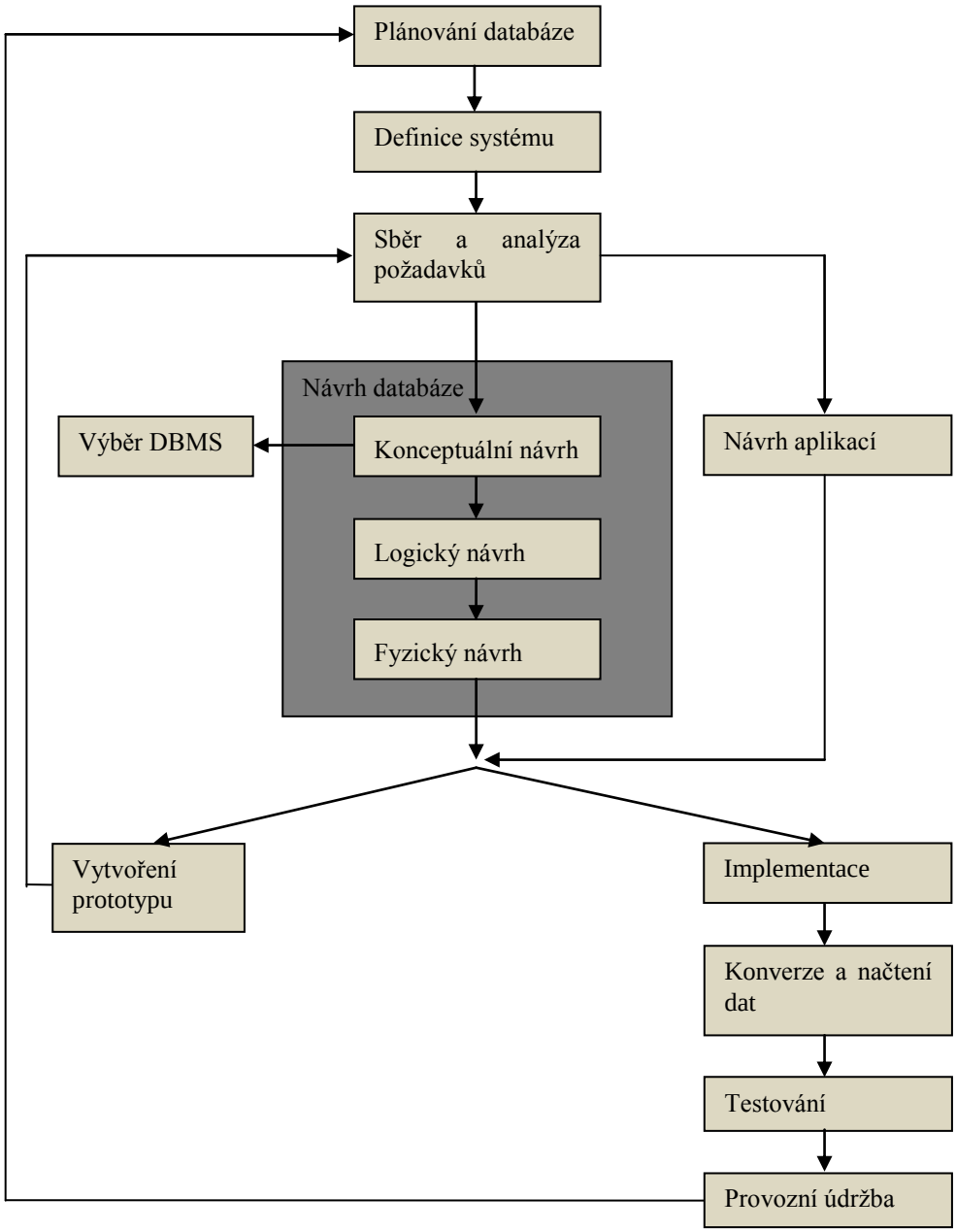
Zdroj: vlastní zpracování

Existují i další způsoby znázornění jednotlivých prvků a jejich kombinace. Při vytváření ER modelu budu v práci využívat první způsob zápisu.

[8] CONOLLY, T. Mistrovství – databáze: Profesionální průvodce tvorbou efektivních databází. 2009. s. 155.

2.2.6 Životní cyklus vývoje databázového systému

Obrázek 2 - Fáze životního cyklu vývoje databázových systémů



Zdroj: [1] s. 110

Na tomto obrázku je znázorněn vývoj databázového systému od plánování managementem až po finální systém a jeho implementaci a údržbu.

2.2.7 Metodologie návrhu databáze

Základem efektivního návrhu složitějších databází je nejprve vytvoření konceptuálního návrhu, potom se z tohoto modelu vytvoří logický návrh a teprve nakonec návrh fyzický.

2.2.7.1 Konceptuální návrh databáze

Hlavním cílem konceptuálního návrhu databáze je vytvoření ER modelu tak, aby při co nejmenší redundanci byly splněny všechny požadavky na databázi. [1]

Krok 1: Identifikace entit – cílem je definování hlavních objektů databáze, o kterých se budou ukládat data.

Krok 2: Identifikace relací – určení vztahů mezi jednotlivými entitami. Nejčastěji se vyskytuje relace mezi dvěma entitami (binární), ale může nastat i vztah mezi více entitami.

Krok 3: Identifikace atributů – identifikace a přiřazení údajů, které budeme v databázi uchovávat, k jednotlivým entitám a relacím.

Krok 4: Určení domén atributů – cílem je určení množiny přípustných hodnot atributů, jejich velikost a formát.

Krok 5: Určení kandidátních a primárních klíčů – identifikace všech možných kandidátních klíčů a výběr toho nejvhodnějšího – primárního klíče.

Krok 6: Specializace/generalizace entit – určení nadtřídy a podtřídy se používá, pokud je třeba rozdělit nějaké entity se shodnými atributy. Tento krok je volitelný.

Krok 7: Kontrola redundance v modelu – pokud se v této fázi v modelu nachází vazba 1:1, tak při identifikaci entit došlo ke zvolení dvou entit pro jeden objekt a proto musíme jednu entitu odstranit.

Krok 8: Kontrola uživatelských transakcí – cílem je zjištění, zda model podporuje všechny transakce, které uživatel od databáze očekává.

Krok 9: Posouzení konceptuálního návrhu databáze s uživateli – představení modelu uživatelům, kontrola a případné provedení změn.

2.2.7.2 Logický návrh databáze

Logický návrh databáze má za úkol v pěti krocích vytvořit z ER modelu (vytvořený v konceptuálním návrhu) tabulky, jejich popis a zkontrolovat strukturu a podporu požadovaných transakcí. [1]

Krok 1: Vytvoření tabulek – cílem prvního a zároveň nejtěžšího kroku této části návrhu má vytvoření tabulek pro entity a relace včetně zanesení všech atributů a integritních omezení a určení primárních a cizích klíčů.

Tabulka 2 - Shrnutí reprezentace entit, relací a atributů s více hodnotami do tabulek

Entita/relace	Mapování
Silná nebo slabá entita	Vytvoření tabulky, která obsahuje všechny jednoduché atributy.
Binární relace 1:*	Umístění kopie primárního klíče entity na straně „jedné“ do tabulky reprezentující entitu na straně „více“. Všechny atributy relace se také umístí na stranu „více“.
Rekurzivní relace 1:*	Protože entity na straně „jedné“ i „více“ jsou stejné, tabulka reprezentující entitu získá kopii primárního klíče, který je přejmenován, a také atributy relace.
Binární relace 1:1 – povinná participace na obou stranách	Spojení entit do jedné tabulky.
Binární relace 1:1 – povinná participace na jedné straně	Umístění kopie primárního klíče entity s nepovinnou participací do tabulky reprezentující entitu s povinnou participací. Všechny atributy relace se umístí také do tabulky reprezentující entitu s povinnou participací.

Binární relace 1:1 – nepovinná participace na obou stranách	Pokud nemáme další informace, umístíme kopii primárního klíče některé entity do tabulky druhé entity. Pokud máme více informací, entitu, která se více blíží povinné participaci, stanovíme jako dceřinou.
Binární relace *:* Komplexní relace	Vytvoření tabulky pro reprezentaci relace a začlenění všech atributů spojených s relací. Umístění kopie primárního klíče z každé rodičovské entity do nové tabulky, kde bude fungovat jako cizí klíč.
Atribut s více hodnotami	Vytvoření tabulek při reprezentaci atributu a umístění kopie primárního klíče rodičovské entity do nové tabulky, kde bude fungovat jako cizí klíč.

Zdroj: [1] s. 242

Krok 2: Kontrola struktury tabulek pomocí normalizace – zjištění, zda jsou všechny tabulky v minimálně 3. normální formě. Pokud ne, musí se provést změny tak, aby tuto podmínku splňovaly.

Krok 3: Kontrola, zda tabulky splňují požadované transakce – ujištění, jestli při vytváření tabulek z ER diagramu nedošlo k chybě a model stále splňuje všechny transakční požadavky uživatele.

Krok 4: Kontrola integritních omezení – cílem je kontrola, zda logický návrh databáze obsahuje všechna integritní omezení. Tím může být definice „Not NULL“ atributů, omezení domén, entitní integrita, referenční integrita, multiplicita a další.

Krok 5: Posouzení konceptuálního návrhu databáze s uživateli – představení modelu uživatelům, kontrola a případné provedení změn.

2.2.7.3 Fyzický návrh databáze

„Proces vytvoření popisu implementace databáze na vnějších paměťových zařízeních; popisuje podkladové tabulky, organizaci souborů, indexy použité kvůli efektivnímu přístupu k datům a všechna související integritní a bezpečnostní omezení.“[9]

Krok 1: Převod logického návrhu databáze do cílového DBMS – tento krok má 3 úkoly. Prvním úkolem je návrh podkladových tabulek pro konkrétní DBMS. Druhý úkol je navrhnout, jak reprezentovat odvozená data, a poslední úkol má zajistit návrh ostatních integritních omezení.

Krok 2: Volba organizace souborů a indexů – zabývá se hlavně výkonností, proto je u menších databází možné tento krok vynechat. Analyzují se prováděné transakce a vybírá se organizace souborů tak, aby databáze byla co nejvýkonnější. V případě potřeby se může zvážit použití indexů (dodatečný klíč).

Krok 3: Návrh uživatelských pohledů – návrh pohledů (např. spojení vybraných sloupců z různých tabulek) podle předem stanovených požadavků uživatele.

Krok 4: Návrh bezpečnostních mechanismů – cílem je zabezpečení databáze pomocí přihlašovacích údajů a zabezpečení přístupu k datům (např. nastavení pravomocí).

Krok 5: Zvážení kontrolovaného zavedení redundance – v tomto kroku by se mělo určit, zda porušení některých pravidel normalizace neprospěje výkonnosti databáze. Opět se řeší pouze u velkých databází náročných na výkon.

Krok 6: Monitorování a vyladění systému v provozu – finální fáze návrhu. Testování rychlosti zpracování požadavků, doby odezvy, ověření funkčnosti uživateli atd. [1]

[9] CONOLLY, T. Mistrovství – databáze: Profesionální průvodce tvorbou efektivních databází. 2009. s. 155.

2.3 SQL (Structured Query Language)

„SQL neboli strukturovaný dotazovací jazyk, byl původně navržen jako databázový jazyk pro komunikaci se SŘBD založenými na relačním modelu. Prvotní verze tohoto jazyka se objevila v polovině 70. let pod názvem SEQUEL a byla vyvinuta firmou IBM jako standardní jazyk pro přístup k relační databázi.

O SQL bychom správně měli 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.“[10]

2.3.1 Funkce SQL

Interaktivní dotazovací jazyk – pomocí příkazů mohou uživatelé ukládat, zobrazovat a pracovat s daty.

Databázový programovací jazyk – začlenění SQL příkazů do aplikačních programů, které mohou být generovány automaticky nebo ovládány uživateli.

Administrační databázový jazyk – oprávněná osoba může přidělovat pravomoci k přístupu k uloženým datům a určit operace, které s nimi daná osoba může dělat.

Jazyk aplikací typu klient/server – sdílená aktuální data jsou uložená na serveru, ale programy, které s nimi pracují a využívají je, jsou přímo v osobním počítači.

Jazyk pro přístup k datům na Internetu – webové aplikace používají SQL pro přístup ke společným datům.

Distribuovaný databázový jazyk – umožňuje rozmisťovat data mezi distribuované databáze a k těmto datům také přistupovat.

Jazyk pro databázové brány – SQL často funguje jako přechod mezi různými databázovými systémy. [3]

---

[10] KŘÍŽ, J. *Databázové systémy.* 2005. s. 15.

2.3.2 Možnosti SQL

*„**Definice dat.** SQL dovoluje uživateli definovat strukturu a organizaci uložených dat spolu s definicí jejich vzájemných vztahů.*

***Získávání dat.** SQL Umožňuje uživateli nebo aplikačnímu programu z databáze uložená data získávat a používat.*

***Manipulace s daty.** SQL umožňuje uživateli nebo aplikačnímu programu databázi aktualizovat přidáváním nových dat, odstraňováním starých nebo změnou již dříve uložených dat.*

***Řízení přístupu.** SQL lze použít k omezení schopnosti uživatele data číst, přidávat a modifikovat, čímž lze chránit před neautorizovaným přístupem.*

***Sdílení dat.** SQL se používá ke sladění sdílení dat mezi více uživateli a k zajištění toho, že se uživatelé mezi sebou navzájem neruší.*

***Integrita dat.** SQL definuje v databázi omezení integrity, čímž ji chrání před porušením, které může vzniknout kvůli neúplným aktualizacím nebo systémovým selháním.“*[11]

2.3.3 Datové typy v SQL

V této kapitole jsou uvedeny některé datové typy, které je možné použít v jazyce SQL.

2.3.3.1 Numerické typy

INTEGER (celé číslo) definice celého čísla (rozsah od -2147483648 do +2147483647).

SMALLINT (celé číslo) definice malého celého čísla (rozsah od -32768 do +32767).

TINYINT (celé číslo) definice malého celého čísla (rozsah od 0 do 255).

NUMERIC (*p,s*) definice desetinného čísla, které má celkem *p* čísel a desetinná čárka je *s* čísel zprava, číslo *s* pevnou desetinnou čárkou.

Jazyk SQL92 povoluje používat místo plných názvů zkratky INT, DEC.

---

[11] GROFF, J., *SQL Kompletní průvodce*. 2005. s. 28-29.

FLOAT (p) definice reálného čísla s plovoucí desetinnou čárkou, parametr *p* udává počet platných číslic (max. 38).

REAL (p) definice reálného čísla s plovoucí desetinnou čárkou, parametr *p* udává počet platných číslic (max. 18).

DOUBLE PRECISION definice reálného čísla s dvojitou přesností (závisí na implementaci).

2.3.3.2 Znakové řetězce

CHARACTER (n) definice řetězce znaků délky *n* (max. 8 000) má-li řetězec méně znaků než *n*, je doplněn zprava prázdnými znaky.

CHARACTER VARYING (n) definice řetězce znaků délky *n* (max. 8 000) - řetězce kratší než *n* se nedoplňují prázdnými znaky.

TEXT (max) – jako VARCHAR - 8 000 bytů.

Jazyk SQL92 povoluje používat místo plných názvů zkratky CHAR, VARCHAR.

2.3.3.3 Typ datum a čas

DATE - definice data ve tvaru *rrrr-mm-dd*.

TIME - definice času ve tvaru *hh:mm:ss*.

SMALLDATETIME – definice času ve tvaru *rrrr-mm-dd hh:mm:ss*.

TIMESTAMP - struktura obsahující datum a čas.

3 Analýza problému a současné situace

Cílem této kapitoly je postavit základ pro úspěšné vytvoření vlastního návrhu řešení. Dále je zde shrnuto testování počítačových programů, které jsou vyvíjeny pro stomatology.

3.1 Popis současné situace

Důsledkem toho, že stomatologové nechtějí, nemůžou nebo prostě jen nevyužívají výpočetní techniku, je už dávno překonané zapisování údajů o pacientech, psaní lékařských zpráv v papírové podobě a zařazování dat podle určitého klíče do kartoték.

3.1.1 Kartotéka – papírová forma databáze

Každému pacientovi se založí „karta“, která má ve skutečnosti podobu jakési obálky, na kterou se nadepíšou údaje o pacientovi (jméno, příjmení, rodné číslo, datum narození, kód pojišťovny…) a do které se vkládají potřebné materiály. Tento způsob uchovávání informací je neskladný a vyhledávání je velice časově náročné. Navíc je poměrně snadné udělat při zápisu a zařazování karty chybu. Další nevýhodou může být změna ošetřujícího lékaře, protože každý, kdo používá papírové databáze, má vlastní způsob zápisu, kterému jiný lékař nemusí přesně porozumět.

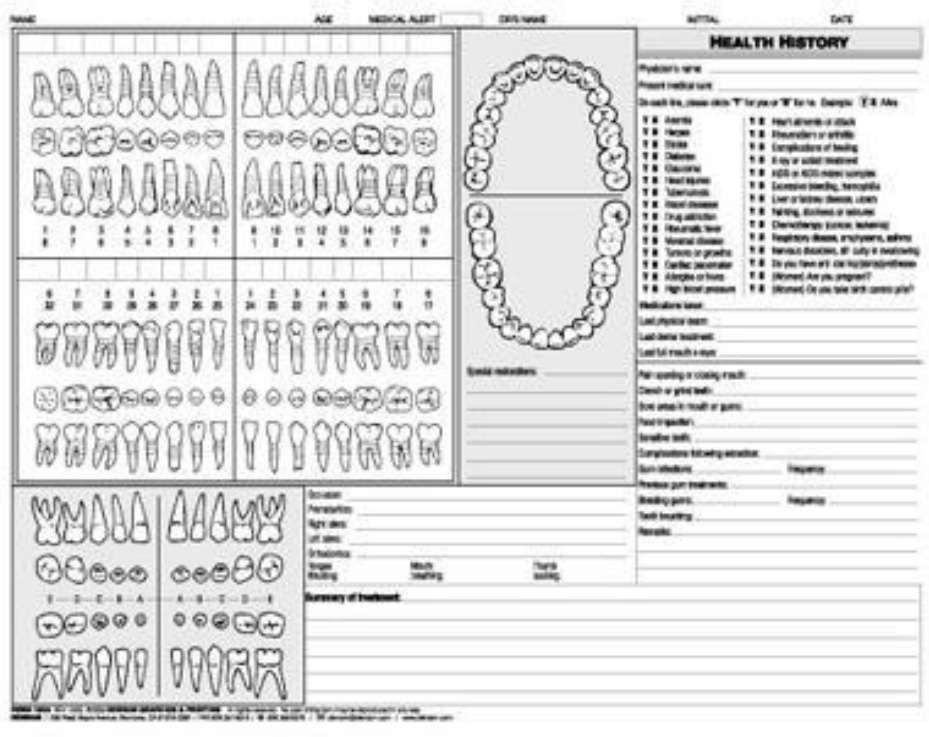
I změna osobních údajů je v „papírových“ databázích částečný problém, protože se údaje musí různě škrtat, přepisovat, přelepovat nebo to končí tím, že se musí celá karta vyplnit znovu. To ovšem zvyšuje už tak vysoké náklady na papírovou databázi. Přitom změny údajů jako jsou adresa, telefonní číslo, zdravotní pojišťovna, u které je pacient pojištěn, jsou na denním pořádku.

3.1.2 Zubní kříž

Zubní kříž zobrazuje aktuální stav jednotlivých zubů pacienta a je základem pro každého stomatologa. Lékař v něm může zpětně najít stavy zubů při jednotlivých prohlídkách, sledovat vývoj změn a při každé prohlídce a zákroku udělat nový zubní kříž tak, aby měl vždy přehled o tom, kdy se pacient zákroku nebo prohlídce podrobil. I zde plynou z využívání papírových kartoték určité nevýhody. Tou hlavní je, že lékař

musí při vypisování nového zubního kříže i při změně stavu jediného zubu opsat předchozí stavy ostatních zubů do nového kříže, což je opět velice časově náročné.

Obrázek 3 - Zubní kříž v papírové podobě



Zdroj: Examination Chart. [online]. 2005 [2010-04-15]. Dostupné na: <http://denram.com/zen/images/101%20S-16.jpg>.

3.1.3 Žádanky o odborné vyšetření

Ve světě moderního lékařství můžeme často slyšet větu: „S tím já vám neporadím, ale pošlu vás k odborníkovi.“ Nemusí se jednat pouze o vyšetření lékařem - specialistou, ale také o různá rentgenová vyšetření. V tomto případě musí ošetřující lékař vypsat žádanku. Žádanky se pro různé účely liší, takže je na lékaři nebo sestře, aby měli v ordinaci vždy dostatek připravených žádanek. Po vyplnění žádanky v papírové podobě ji sám pacient musí odnést k lékaři, který u něj bude vykonávat speciální zákrok, nebo na oddělení RTG. Často se pacient dozví, že ho teď neošetří a že se musí objednat, nebo že výsledky testů či rentgenu nebudou hotové na počkání. Až když jsou výsledky připravené, musí je pacient vyzvednout a odnést zpět svému ošetřujícímu lékaři. Tím získává roli nejen pacienta, ale i jakéhosi doručovatele. To je náročné na

pacientův čas, ale také se může stát, že se žádanka nebo už výsledky testů či rentgenový snímek ztratí, a tak se musí pacient pustit do celého koloběhu znovu.

3.1.4 Objednání pacienta

Překvapivě mnoho lékařů stále využívá k objednávání pacientů obyčejný diář. Proces objednání je pak založen na pacientově hádání termínů, které by mohl mít lékař volné, dokud se nenalezne shoda. To často spěje k tomu, že pacient kývne i na termín, který by si za normálních okolností nevybral, a za 2 hodiny, když se do svého diáře podívá v klidu, telefonuje do ordinace, že se mu vybraný termín nehodí a celý proces objednání se opakuje.

Protože pravidelné prohlídky u stomatologa by měly probíhat minimálně dvakrát za rok, musel by se pacient při návštěvě objednávat na půl roku dopředu. Pak se často stane, že pacient na prohlídku zapomene. To se v mnoha případech řeší tím, že lékař pacientovi při návštěvě řekne, ať přijde na další prohlídku za půl roku, ale že se musí objednat dva měsíce předem. I v tomto případě nastávají určité problémy, např. pokud si pacient vzpomene až po půl roce, že se měl objednat. Potom se doba mezi jednotlivými prohlídkami prodlužuje z původních šesti měsíců na osm nebo devět měsíců.

3.1.5 Poplatky od pacienta

I před zavedením regulačního poplatku se ve stomatologii nacházely určité zákroky, které byly pojišťovnou hrazeny pouze částečně nebo vůbec. Jsou to příplatky za bílou plombu, injekce proti bolesti, bělení zubů atd. Tyto zákroky pacient hradí přímo v ordinaci. Lékař nebo sestra musí ohlídat, na které z provedených zákroků se vztahuje doplatek (často závisí na pacientově zdravotní pojišťovně – jiná výše hrazené částky za stejné zákroky u různých zdravotních pojišťoven), a pak tyto zákroky zvlášť evidovat, vypisovat potvrzení o zaplacení a kontrolovat finanční hotovost tak, aby na konci dne souhlasila s evidovanými zákroky.

3.1.6 Dokumentace pro zdravotní pojišťovny

Nejobsáhlejším výstupním dokumentem lékařů je seznam provedených zákroků všem pacientům, který slouží jako výkaz pro zdravotní pojišťovny. Výkaz se posílá jednotlivým zdravotním pojišťovnám a slouží jako podklad k proplacení všech pojišťovnou plně nebo částečně hrazených zákroků.

Jednotlivé zákroky, které lékař pacientovi provede, musí sestra zaznamenat nejen do pacientovy karty, ale také do výkazu pro zdravotní pojišťovnu, u které je pacient pojištěn. To často prodlužuje dobu ošetřování pacienta nebo zvyšuje dobu mezi ošetřením jednotlivých pacientů, kdy sestra přepisuje zákroky z karty pacienta do výkazu pro zdravotní pojišťovny.

3.2 Analýza problému

Na základě popisu současné situace lze poukázat na základní body, jejichž zlepšení by mělo být stěžejní v samotném návrhu databáze. Většinu nedostatků, na které bylo upozorněno v předchozí kapitole, je možné odstranit použitím elektronické relační databáze. Navíc se ušetří mnoho času a nákladů na zbytečné „papírování“.

3.2.1 Vztah pacient – lékař (sestra)

Vytvoření karty pacienta v elektronické podobě by mělo spočívat v nějakém formuláři, do kterého sestra vypíše údaje o pacientovi. Změna osobních údajů pak nebude činit žádný problém. K této kartě se pak budou přiřazovat jednotlivé zákroky, obrázky RTG atd. tak, jak to bylo v papírové kartotéce.

U zubního kříže bude k jednotlivým zubům umožněn výběr všech stavů, které mohou u pacientů nastat. U zubního kříže bude uvedeno i datum tak, aby bylo možné zpětně dohledávat jednotlivé záznamy. Pokud bude lékař chtít vytvořit nový zubní kříž pacienta, otevře se mu pacientův poslední vytvořený kříž, aby mohl lékař pouze zaznamenat změny a nemusel zbytečně vyplňovat kolonky i u těch zubů, jejichž stav zůstal nezměněn.

Žádanky o odborné vyšetření, stejně jako jeho výsledky, budou zasílány systémem lékaři – specialistovi, resp. přiřazeny ke kartě pacienta. Tím se urychlí předávání žádanek a hlavně se ušetří čas pacienta při přebírání výsledků. Stejně tak to bude i u rentgenových vyšetření, kdy lékař pošle elektronickou žádost přímo na oddělení RTG a oni naopak pošlou elektronickou formou rentgen přímo lékaři.

Objednání pacienta se elektronickou formou databáze také zlepší, a to jak z pohledu lékaře, tak i z pohledu pacienta. Databáze může být napojena na webové stránky kliniky (ordinace), kde se budou aktuálně zobrazovat volné a obsazené termíny zubního lékaře, a tak si pacient bude moci vybrat, než zavolá do ordinace a objedná se. Bohužel není možné, aby se pacienti na internetu objednávali sami elektronicky, protože je důležité, aby sestra podle informací od pacienta rozhodla, jak dlouho bude prohlídka a případný zákrok trvat. Jinak by se mohlo stát, že lékař bude mít mezi jednotlivými pacienty prodlevy nebo že naopak u některého pacienta doba zákroku přesáhne dobu, na kterou

byl objednán, a tím způsobí určité „zpoždění“, které je nepříjemné pro ostatní pacienty v čekárně.

Další výhodou databáze bude automatické připomínání. Využití elektronické databáze umožní odesílání upozornění pacientům, kteří ve své kartě mají vyplněný e-mail. Například je možné upozornit pacienty 3 dny předem na objednaný termín nebo 90 dní (3 měsíce) po jejich poslední návštěvě připomenout, že by se měli objednat na další kontrolu.

U provedených zákroků bude systém automaticky upozorňovat, že pacient musí za zákrok nějakou částku uhradit. Tím lékaři (sestře) odpadne nutnost kontroly, zda zákrok plně hradí pacientova zdravotní pojišťovna nebo ne.

3.2.2 Vztah lékař – zdravotní pojišťovna

Jakmile lékař zapíše každý provedený zákrok do databáze, nebude už problém pomocí pohledů tisknout sestavy a výkazy pro jednotlivé zdravotní pojišťovny. Tím se ušetří dvojí zapisování zákroků, které bylo nezbytné při využívání papírových kartoték.

3.3 Popis vybraných informačních systémů pro stomatology

V následující kapitole je shrnutí některých IS pro stomatology. Tyto profesionální programy jsou velmi drahé a pro člověka se základní počítačovou gramotností poněkud komplikované. Systémy většinou nejsou volně dostupné a většinu demoverzí je možné vyzkoušet pouze na požádání. O některé programy jsem zažádal, ovšem pouze společnost Sakura mi poslala svou demoverzi programu Dentist+. Dále jsem vyzkoušel volně dostupné demoverze programů DentiMax a Dentasoft.

3.3.1 Sakura Dentist+

Po vyplnění jednoduchého formuláře na stránkách společnosti mi během dvou týdnů přišla 90-ti denní zkušební verze toho programu. Instalace jsem odzkoušel na operačním systému Windows Vista. První problém nastal již při instalaci softwaru. Protože si program neumí poradit s ochranou systému, je nutné buď vypnout ochranu systému, nebo složitě nastavovat program, aby ho systém při každém spuštění spouštěl jako administrátor.

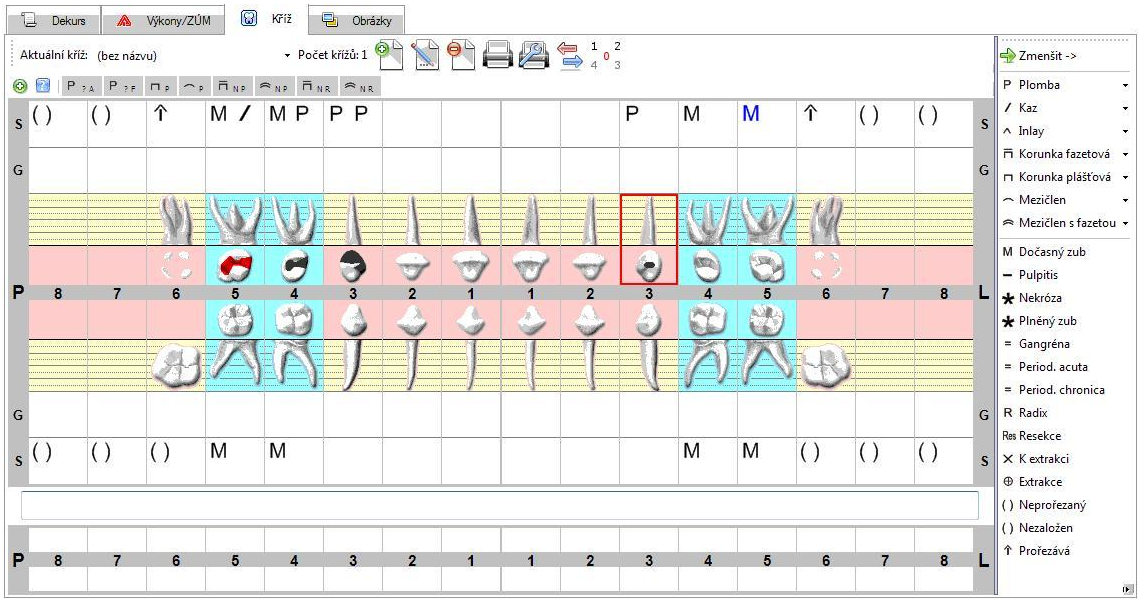
Program umí vše, na co by si mohl stomatolog vzpomenout, možná i něco navíc. Umožňuje založit zdravotnické středisko, přidat do něj nové lékaře a získávat souhrnné informace o všem, co lékaři udělali. Tato funkce je vhodná pro větší kliniky, které mají vlastní vedení.

Aplikace nabízí všechny obvyklé funkce jako karta pacienta, objednání pacienta, zubní kříž, dekurs (lékařský zápis) atd. Program nabízí i možnost prodeje vlastních výrobků, vytváření vlastních léčebných postupů, správu materiálu, seznam kontaktů obchodních partnerů a mnoho dalších. Vše je dokonalé a možná až přehnané. Zbytečný mi přišel například věk pacienta při zobrazení pacientovy karty ve formátu: „VĚK: 57 let, 4 měsíce, 7 dnů“. Takových funkcí je zde dle mého názoru mnoho. Pokud by chtěl lékař nahradit svou papírovou databázi tímto programem, strávil by několik dní až týdnů pouze učením se práce s programem.

Zubní kříž je zde hezky graficky zpracován, ale opět je trochu přehlcen potřebným nastavováním. Například pokud lékař pacientovi udělá plombu, musí do zubního kříže zanést i to na jakém místě na zubu se plomba nachází.

Program dále vytváří poměrně přehledné sestavy (výkazy) pro pojišťovny a umožňuje je přímo odesílat pojišťovně. Další výhodou je připojování rentgenových snímků z jakýchkoliv zařízení, která jsou připojena k počítači, takže je možné z moderních rentgenových zařízení, která mají digitální výstup, nahrávat rentgeny přímo do PC.

Obrázek 4 - Zubní kříž Sakura Dentist+



Zdroj: screenshot

Kromě cen uvedených v následující tabulce si společnost účtuje cestovné 10 Kč/km. Pokud si lékař nezaplatí službu Podpora, operativní servis stojí 1 500 Kč/hod. I s touto službou stejně zaplatí 750 Kč/hod a za radu po telefonu zaplatí až 400 Kč/15 minut. Pokud nechce, aby jeho verze obsahovala komerční sdělení, zaplatí o dalších 50 % navíc. Aktualizace jsou dostupné pouze pro předplatitele služby Podpora. Všechny ceny jsou uvedeny bez DPH 20 %.

Obrázek 5 - Ceník Sakura Dentist+

	Licence	Instalace a zaškolení [1]	Podpora 12 měsíců [2]
Dentist+ základní pracoviště	14.500	2.000	4.860
Rozšíření o další systém Dentist+ další systém k provozování samostatně nebo k napojení do počítačové sítě	6.900	500	1.290
Modul **Správce obrazové dokumentace** základní pracoviště	4.500	-	990
Modul **Správce obrazové dokumentace** další pracoviště	990	-	198
Přechod na novou generaci programu			
Základní pracoviště	4.999	2.000	-
Další pracoviště	1.499	500	-

[1] Cestovné **není** v ceně instalace.
[2] Verze neobsahující komerční sdělení (reklamu) má cenu předplatného uživatelské podpory navýšenu o 50%.

Zdroj: Ceník programu Dentist+. [online]. [2010-04-15]. Dostupné na: <http://dentist.cz/cenik.html>.

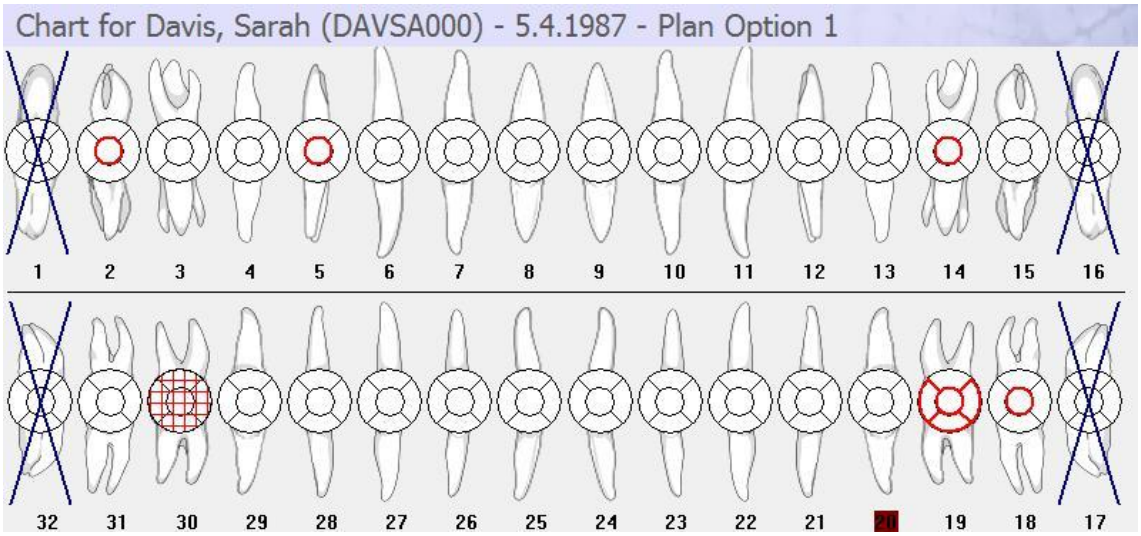
Využívání tohoto programu po dobu pěti let na klinice s deseti lékaři tak, aby měli software aktualizovaný, bez reklam a včetně modulu Správce obrazové dokumentace, by stálo 288 900 Kč včetně DPH. Tato cena je poměrně vysoká, ale cena na uživatele prudce klesá při vyšším počtu uživatelů.

3.3.2 DentiMax

Další z testovaných softwarů je program DentiMax. Demoverzi bylo možno stáhnout na internetu, i když samotné internetové stránky firmy tuto možnost nenabízí. DentiMax funguje bezproblémově ve Windows Vista. Program pochází od zahraniční společnosti a tak je kompletně v anglickém jazyce, což může být první mírný nedostatek pro české uživatele. Program je poměrně pomalý – uživatel musí mezi načtením každé obrazovky čekat okolo 1 vteřiny – to je při delší práci s programem nepříjemné.

Software splňuje většinu základních požadavků – obsahuje kartu pacienta, objednávání pacientů, dekurs, zubní kříž, speciální vyšetření (např. periodontální), statistiky, výkazy. Program je opět velmi složitý a nepřehledný. Umožňuje tisk až 20 různých žádanek, potvrzení a formulářů, které se pak vypisují v papírové podobě.

Obrázek 6 - Zubní kříž DentiMax



Zdroj: screenshot

Zubní kříž je znázorněn graficky. Vytknout mu lze to, že při zadávání stavu zubu musíte vybírat z mnoha podobných obrázků místo klasické textové nabídky.

Plná verze programu stojí 2 349 $ pro jediného uživatele a multilicence pro 8 uživatelů stojí 4 249 $. Tento program nejen kvůli anglickému jazyku, ale i celkovému zpracování prostředí, je mnohem méně uživatelsky příjemný než software Dentist+.

3.3.3 Dentasoft

Poslední ze stomatologických programů, které se mi podařilo získat, je software Dentasoft společnosti Tomšů Software s.r.o. Společnost sice vyvíjí ve spolupráci s firmou CompuGROUP Česká republika s.r.o. také moderní program PC DENT, ale neposkytuje uživatelům možnost stažení nějaké demoverze. Program Dentasoft se mi na Windows Vista ani Windows XP nepodařilo vůbec zprovoznit i přesto, že je autor uvádí v podporovaných operačních systémech. Při spuštění demoverze program zažádá o zadání hesla, které není uvedené přímo na internetové stránce, ale je třeba hledat v adresáři, v kterém je software nainstalován, textový soubor s nastavením programu. Potom program požádá o potvrzení, resp. nastavení aktuálního data a času – pak nahlásí chybu. Prostředí tohoto programu se za posledních patnáct let vůbec nezměnilo. Tento software je vytvářen nad zastaralým operačním systémem DOS.

Z internetových stránek jsem zjistil, že program sice umí většinu požadovaných funkcí, nicméně při pohledu na následující obrázek je zřetelné, že prostředí je opravdu zastaralé a nepříjemné.

Obrázek 7 - Zubní kříž Dentasoft

Zdroj: Zubní kříž. [online]. 2008 [2010-04-16]. Dostupné na: <http://www.tomsusoftware.cz/dentasoft#>.

Jedna licence programu 9 800 Kč a za roční podporu musí uživatel zaplatit 3 060 Kč. Tato cena je za takto zastaralý systém opravdu přehnaná. Modernější program PC DENT stojí 17 800 Kč a 5 190 Kč za podporu. Uživatel si k němu může koupit také Modul obrazový archiv snímků a dokumentace za 5 900 Kč a Modul objednací kalendář za 600 Kč. Všechny tyto ceny jsou uvedeny bez DPH 20 %.

4 Vlastní návrhy řešení, přínos návrhů řešení

Na základě analýzy problému, popisu současné situace a základních znalostí problematiky jsem byl schopný pokročit ve vlastním návrhu řešení databáze.

4.1 Zachycení požadavků a procesů

Tuto fázi jsem rozdělil na tři části – hlavní proces Běžný chod ordinace a subprocesy Registrace pacienta a Objednání pacienta. Pomocí vývojových diagramů jsem zachytil většinu požadavků a procesů, které by se měly v databázi vyskytovat.

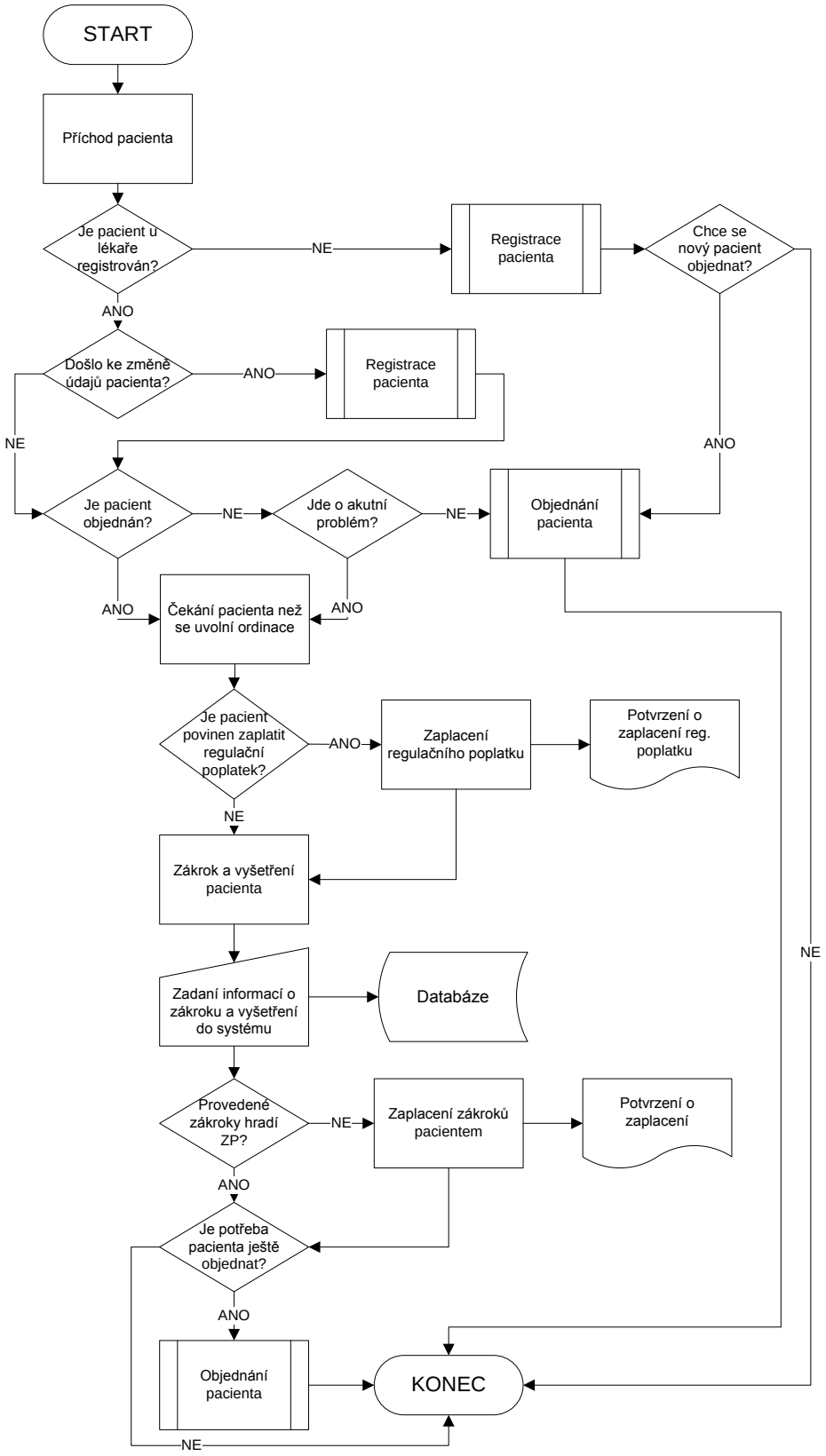
Některé další možnosti databáze, vzniklé z požadavků zjištěných při popisování současné situace a analýze problému, nejsou ve vývojovém diagramu zaneseny a zařadím je až v logickém návrhu databáze. Jedná se například o žádanky na odborná vyšetření, žádanky o RTG vyšetření atd.

4.1.1 Hlavní proces Běžný chod ordinace

V hlavním procesu jsem zachytil základní činnosti, které nastanou nebo mohou nastat po příchodu pacienta do ordinace. Systém musí umožnit lékaři nebo sestře zaregistrování, kontrolu údajů a objednání pacienta.

Dále se ověřují podmínky, jestli je pacient objednán či jde o akutní problém. Databáze musí obsahovat možnost zaplacení regulačního poplatku nebo doplatku za nadstandardní zákroky pacienta, u kterých musí částečnou nebo celou částku hradit pacient.

Obrázek 8 - Vývojový diagram – Běžný chod ordinace

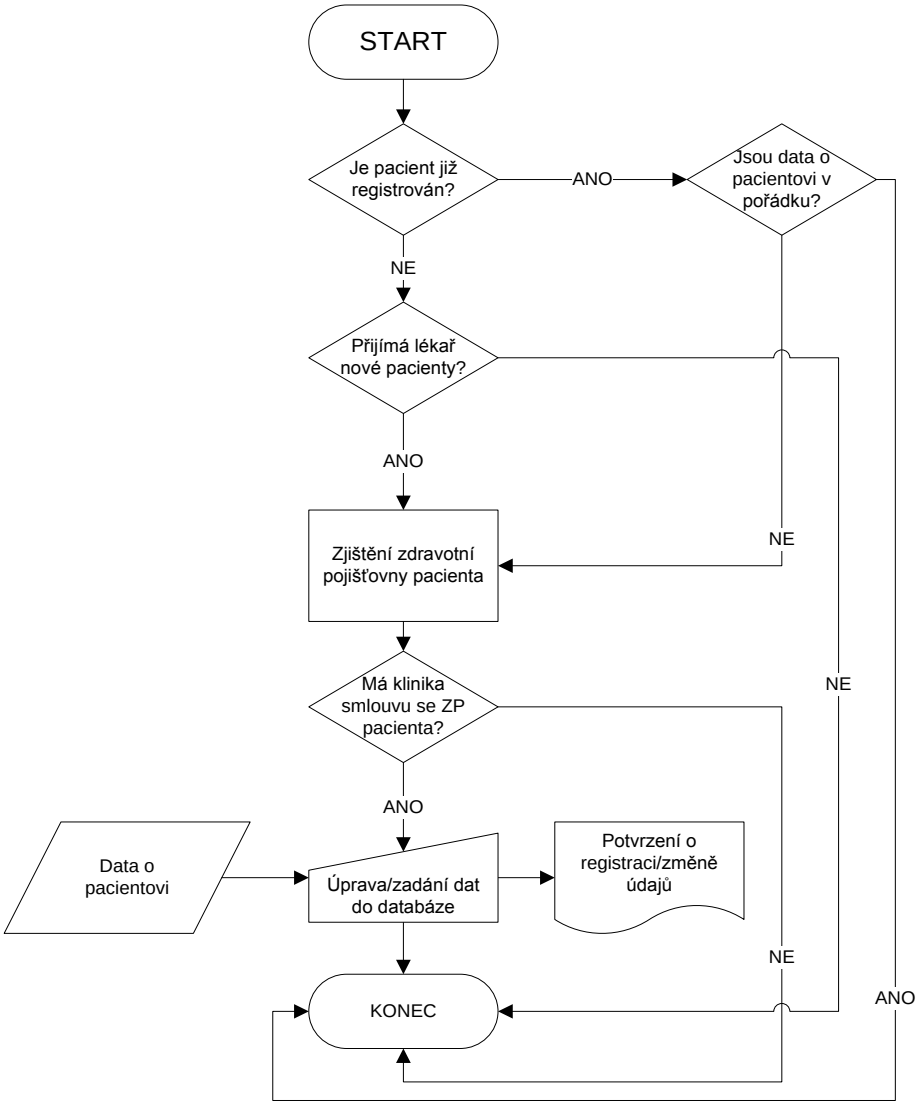


Zdroj: vlastní zpracování

4.1.2 Subproces Registrace pacienta

Při vstupu do toho subprocesu je nutné ověřit hned několik podmínek. Nejdříve je nutné ověřit, zda není pacient už jednou zaregistrován. Tím se zjistí, jestli se do tohoto procesu vstoupilo za účelem nové registrace nebo kvůli změně údajů o pacientovi. Dalším z úkolů, které musí být splněny, je zjistit, zda lékař stále přijímá nové pacienty. Poslední z podmínek se týká zdravotních pojišťoven. V poslední době stále přibývá lékařů, kteří nemají smlouvu s některými pojišťovnami, a tak se musí ověřit, zda s pacientovou zdravotní pojišťovnou má daný lékař smlouvu. Jinak by totiž zákroky nebyly lékaři proplaceny. Po splnění těchto podmínek už nic nebrání zadaní dat.

Obrázek 9 - Vývojový diagram – Registrace pacienta

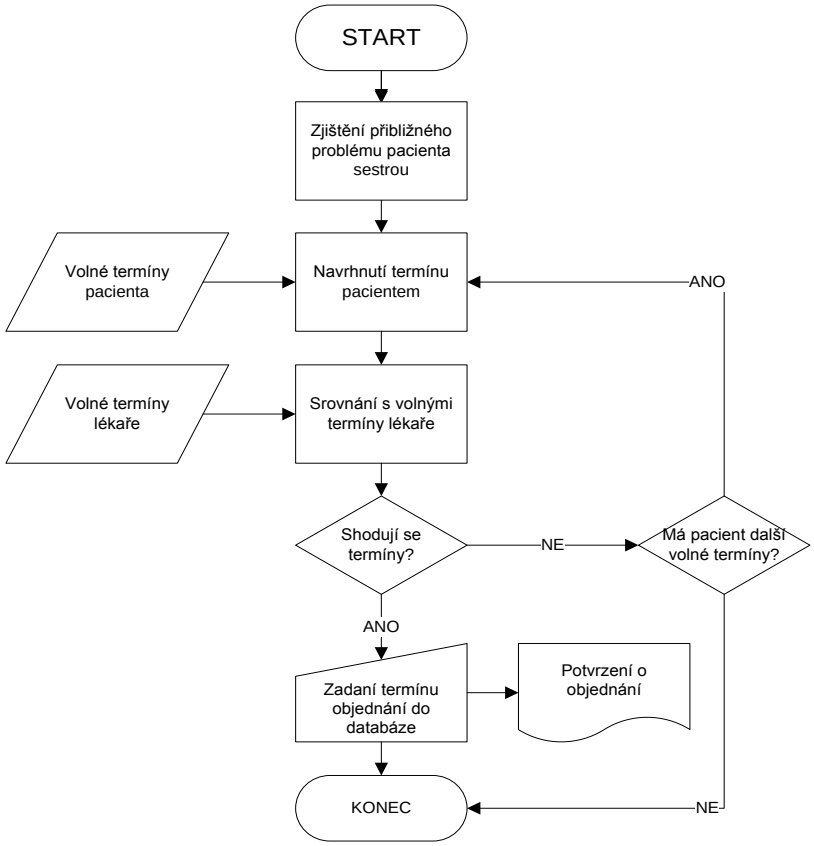


Zdroj: vlastní zpracování

4.1.3 Objednání pacienta

Proces objednání funguje téměř stejně, jak je popsáno v analýze problému. Až na to, že ve výsledku bude mít pacient možnost sledovat volné termíny na internetových stránkách.

Obrázek 10 - Vývojový diagram – Objednání pacienta



Zdroj: vlastní zpracování

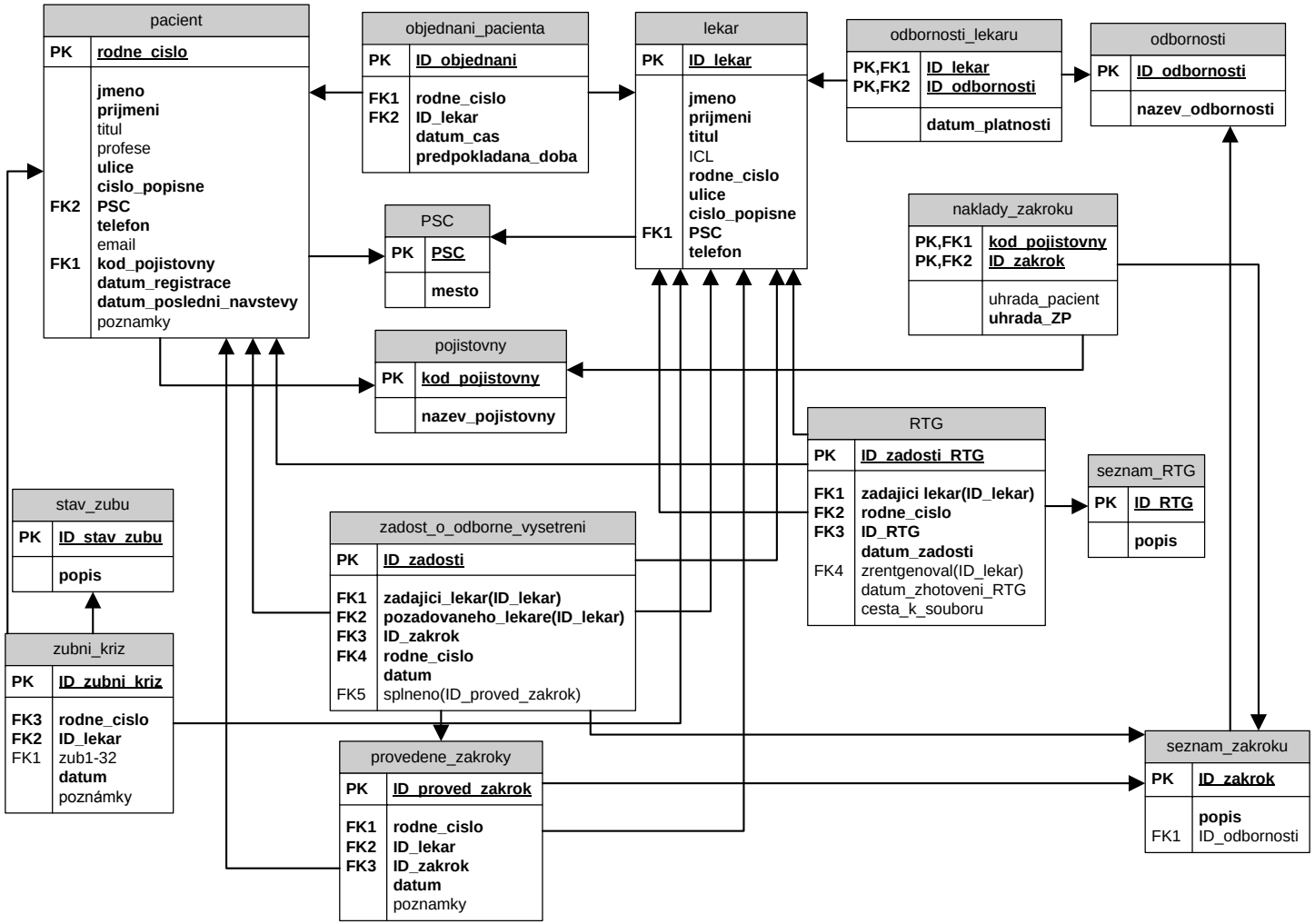
4.2 Konceptuální návrh

Cílem této části je vytvoření ER modelu tak, aby při co nejmenší redundanci byly splněny všechny požadavky na databázi. Postupuje se od identifikace entit, relací a atributů přes určení kandidátních klíčů až ke kontrole redundance v modelu a posouzení celého návrhu.

4.3 Logický návrh

Logický návrh vytvoří z entit tabulky, zkontroluje strukturu pomocí normalizace, ověří požadované transakce a integritní omezení.

Obrázek 11 - Logický návrh databáze



Zdroj: vlastní zpracování

Logický návrh obsahuje všechny entity (tabulky), které bude databáze obsahovat. V návrhu jsou zobrazeny relace, primární i cizí klíče a tučným písmem jsou zobrazeny atributy, které musí být vyplněny.

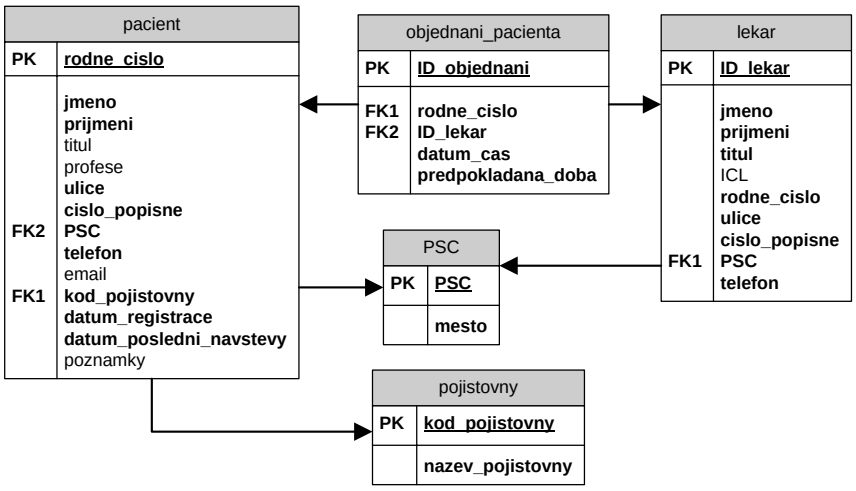
4.3.1 Pacient, lékař a objednání pacienta

Tabulka *pacient* je určena primárním klíčem vytvořeným z rodného čísla a obsahuje základní údaje o pacientovi – karta pacienta. Pokud bude u pacienta vyplněn *email,* budou mu automaticky zasílána upozornění na objednaný termín a upozornění, že by se měl objednat. Kód pojišťovny musí být vyplněn a je možné vybrat ho pouze z tabulky *pojistovny*, která bude obsahovat pouze nasmlouvané pojišťovny. Tím je ošetřeno riziko vyšetření pacienta pojišťovny, s níž nemá lékař smlouvu. Datum poslední návštěvy pacienta je sice redundantní položkou, ale výrazně urychlí pozdější vyhledávání v databázi.

Tabulka *lekar* je určena primárním klíčem *ID_lekar* a jsou v ní opět uvedeny základní údaje o lékaři. *ICL* je identifikační číslo lékaře, které nemusí být vyplněno, protože se v poslední době přestává využívat.

Tabulka *Objednani_pacienta* obsahuje jedinečný identifikátor ID a ukazuje, který pacient je objednán, k jakému lékaři a na kdy. Položka doba bude obsahovat sestrou odhadnutou dobu trvání zákroku, aby nedocházelo ke zbytečným prodlevám. Z této tabulky by poté mohla vycházet internetová stránka s přehledem volných termínů.

Obrázek 12 - Objednání pacienta



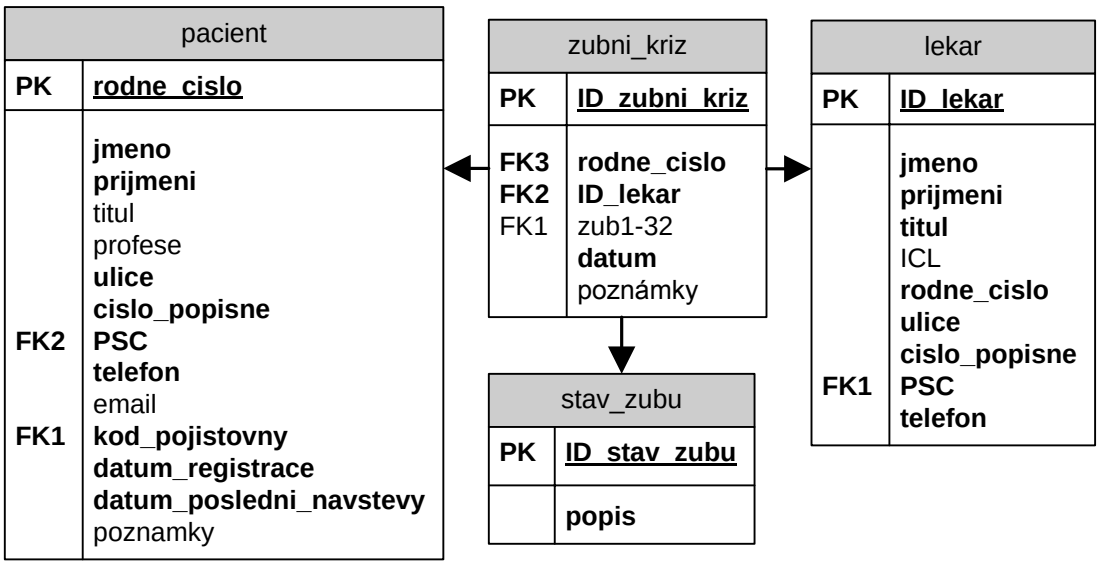
Zdroj: vlastní zpracování

4.3.2 Zubní kříž

Zubní kříž obsahuje automaticky generovaný primární klíč, rodné číslo ošetřovaného pacienta, ID ošetřujícího lékaře a datum, kdy byl zubní kříž vytvořen. Toto datum se po vytvoření kříže automaticky uloží také do data poslední návštěvy pacienta. Položka *zub1-32* se ve fyzickém návrhu databáze bude opakovat 32 krát (*Zub 1, Zub 2, atd.*). Ke každému zubu tak bude možné vybrat jednu možnost z tabulky *stav_zubu*, kde budou zaznamenané všechny stavy, které mohou nastat. Lékař má možnost si zapsat ke kříži poznámky.

Při vytváření nového zubního kříže pro pacienta se načte jeho předcházející kříž tak, aby lékař nemusel vyplňovat všechny stavy zubů znovu. Staré zubní kříže však zůstanou uloženy, aby bylo možné sledovat změny v stavu pacientova chrupu.

Obrázek 13 - Zubní kříž

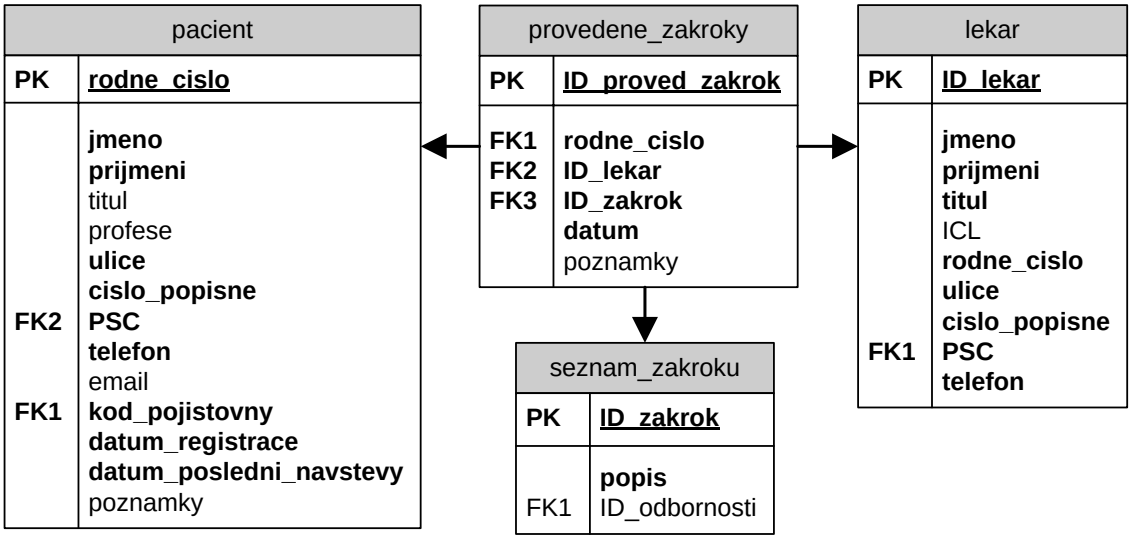


Zdroj: vlastní zpracování

4.3.3 Provedené zákroky

Provedené zákroky jsou určeny primárním klíčem *ID_proved_zakrok* a opět obsahují rodné číslo pacienta, ID lékaře a datum provedení zákroku. Další cizí klíč je zde *ID_zakrok*, který provedené zákroky spojuje s tabulkou *seznam_zakroku*. V ní jsou obsaženy všechny nabízené zákroky, které lékař provádí a jejich popis. Dále jsou v ní uvedeny odbornosti, o kterých se zmíním později.

Obrázek 14 - Provedené zákroky

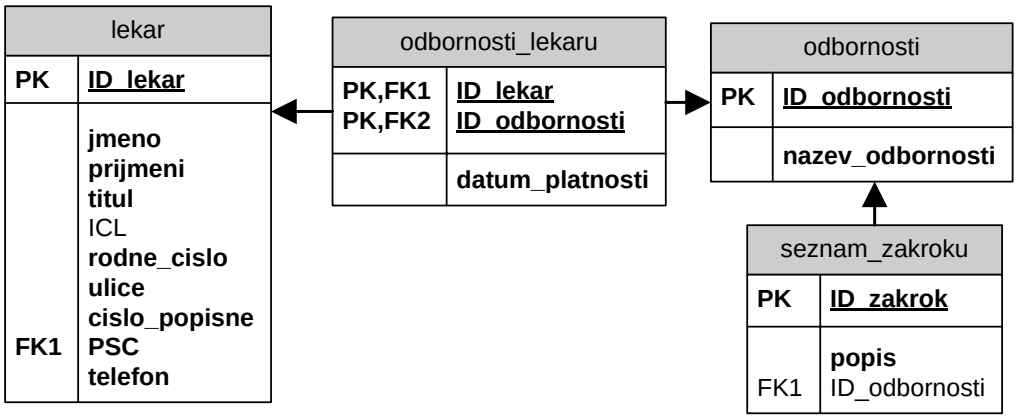


Zdroj: vlastní zpracování

4.3.4 Odbornosti lékařů

Na některé zákroky musí mít lékař určitou specializaci – odbornost. V *seznam_zakroku* je u zákroků vyžadující odbornost vyplněno *ID_odbornosti* z tabulky *odbornosti*, jinak je položka prázdná. V tabulce *odbornosti_lekaru* jsou složenými primárními a zároveň cizími klíči *ID_lekar* a *ID_odbornosti* a je v ní uvedeno, jaké odbornosti má každý lékař. Dále je zde *datum_platnosti*, po jehož překročení se záznam v tabulce automaticky smaže. Odbornost totiž lékař nezískává na celý život, ale pouze na 3 nebo 5 let. Toto spojení tabulek slouží k zamezení vykonávání zákroků lékařem, který nemá potřebnou odbornost.

Obrázek 15 - Odbornosti lékařů

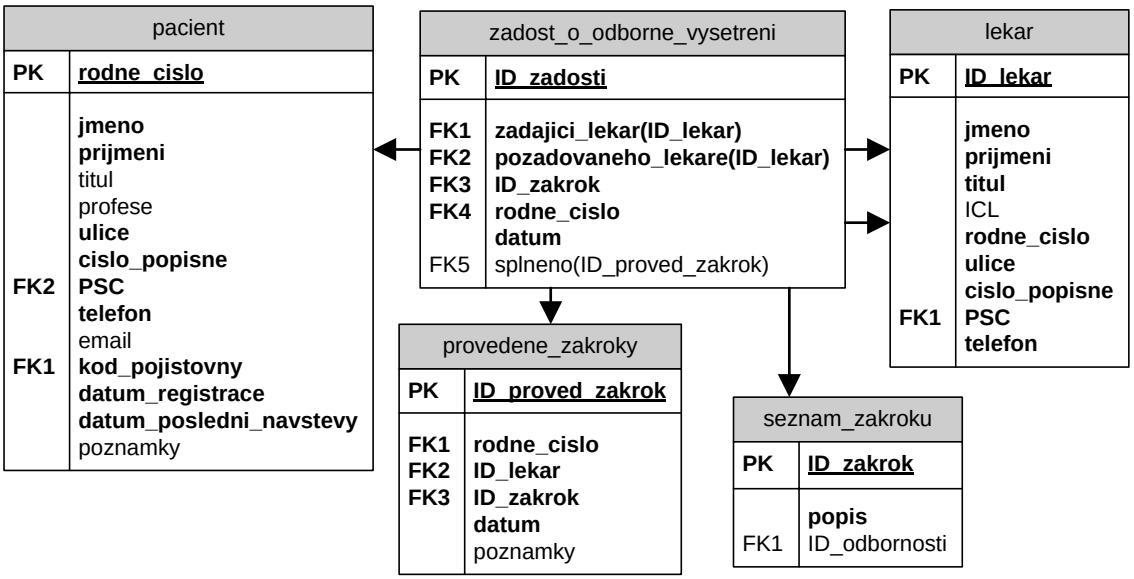


Zdroj: vlastní zpracování

4.3.5 Žádost o odborné vyšetření

V žádosti o odborné vyšetření je uvedeno ID lékaře, který žádá, a ID lékaře, který je žádán. Dále je zde uvedeno datum, rodné číslo pacienta a ID zákroku, který má pacient u žádaného lékaře absolvovat. Po provedení vyšetření nebo zákroku ho lékař, který zákrok prováděl, zapíše do tabulky *provedene_zakroky* a ID tohoto zákroku zapíše do kolonky *splneno* u dané žádosti. Tím se žádost považuje za vyřízenou.

Obrázek 16 - Žádost o odborné vyšetření

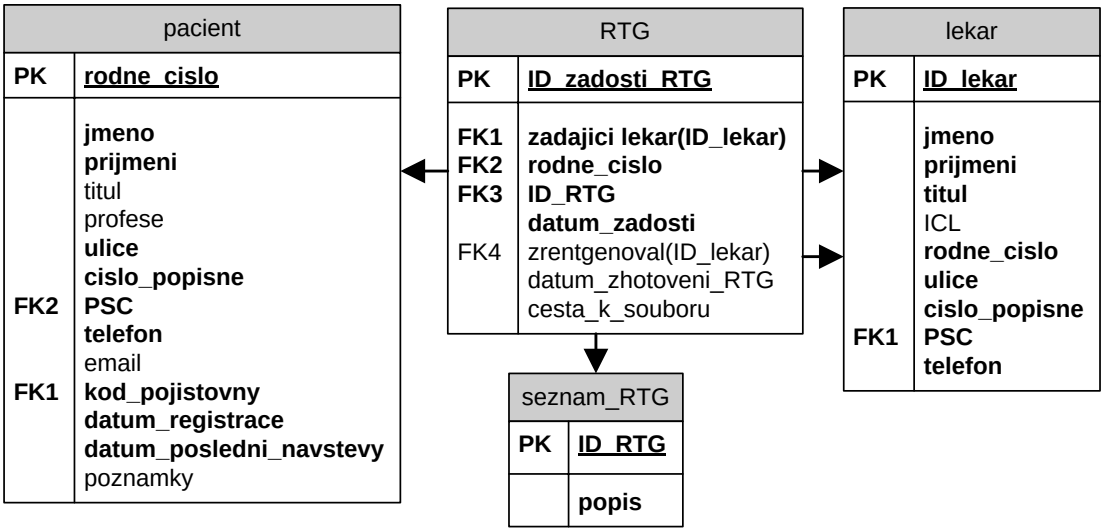


Zdroj: vlastní zpracování

4.3.6 Žádost o RTG

Žádost o RTG je podobná jako žádost o odborné vyšetření. Pouze se nevybírá z tabulky zákroků, ale z tabulky *seznam_RTG*, kde je seznam všech nabízených rentgenových snímků. Po provedení se k žádosti připojí ID lékaře, který rentgen zhotovil, datum a elektronická cesta, na které je soubor s rentgenem k dosažení.

Obrázek 17 - Žádost o RTG

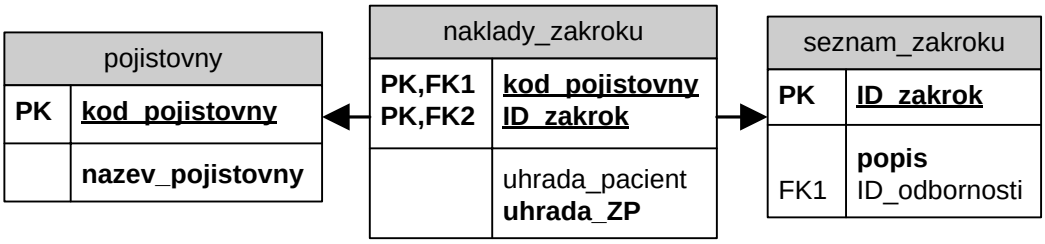


Zdroj: vlastní zpracování

4.3.7 Náklady zákroku

Poslední kombinací tabulek se řeší náklady zákroků. V tabulce je uvedeno, jakou částku za zákrok hradí pojišťovna a jakou pacient. Výše těchto částek se liší u jednotlivých pojišťoven, a tak není možné přiřadit částky přímo k zákroku, ale náklady musí být řešeny tímto způsobem.

Obrázek 18 - Náklady zákroků



Zdroj: vlastní zpracování

4.4 Fyzický návrh databáze

V této kapitole přesněji popíšu některé procedury, triggery a pohledy, které jsou v databázi použity. Model fyzického návrhu databáze, který obsahuje i datové typy a jejich délky, zobrazuje příloha č. 1. Datový slovník se nachází v příloze č. 2. Celý zdrojový kód obsahující vytvoření všech tabulek, příklady insertů (vložení dat do tabulky), všechny použité procedury, triggery a pohledy je uveden v příloze č. 3.

4.4.1 Procedura – Prověř odbornosti

Tato procedura s kurzorem má jeden vstupní parametr – aktuální datum. Jejím cílem je kontrola platnosti odborností jednotlivých lékařů v tabulce *odbornosti_lekaru*. V této tabulce porovnává data uvedená v sloupci *datum_platnosti* a srovnává je se vstupním parametrem. Pokud je datum platnosti menší (starší) než aktuální datum, celý řádek se smaže. Lékař potom nebude moci přidávat do systému zákroky, které vyžadují odstraněnou odbornost, což je ošetřeno další procedurou.

```
create procedure prover_odbornosti
      @dat1 smalldatetime
as
begin
      declare
      @id_odbor tinyint,
      @id_lek smallint,
      @dat2 smalldatetime

declare cur_odbor cursor for
      select o.ID_odbornosti, o.ID_lekar, o.datum_platnosti
      from odbornosti_lekaru o
open cur_odbor
      fetch next from cur_odbor into @id_odbor, @id_lek, @dat2
while @@fetch_status=0
      begin
if (@dat2<@dat1)
      begin
delete from odbornosti_lekaru where (ID_odbornosti=@id_odbor) and
(ID_lekar=@id_lek)

      end
      fetch next from cur_odbor into @id_odbor, @id_lek, @dat2
      end
close cur_odbor
deallocate cur_odbor
end
```

Dotaz na spuštění procedury:

```
exec prover_odbornosti '2010-04-15'
```

4.4.2 Procedura – Přidej zákrok

Tato procedura slouží k ověření, zda má lékař na provedení zákroku požadovanou odbornost. Po zadání vstupních parametrů *rodne_cislo, ID_lekar, ID_zakrok, datum* a *poznamky* se procedura „podívá“, jestli zadaný zákrok vyžaduje nějakou odbornost (tabulka *seznam_zakroku*) a pokud ano, zda zadaný lékař má tuto odbornost uvedenou v tabulce *odbornosti_lekaru*. Pokud tento záznam nenajde, neumožní lékaři tento zákrok do databáze přidat. V případě, že zákrok nevyžaduje žádnou odbornost, může záznam vytvořit jakýkoliv lékař.

```
create procedure pridej_zakrok
  @rodne_cislo numeric(10),
  @ID_lekar smallint,
  @ID_zakrok smallint,
  @datum smalldatetime,
  @poznamky text

as
begin
      declare
      @pom_odbor tinyint
      set @pom_odbor = (select ID_odbornosti from seznam_zakroku where
      ID_zakrok=@ID_zakrok)

if ((@pom_odbor is NULL) or (select count(ID_odbornosti) from
      odbornosti_lekaru
      where ID_lekar=@ID_lekar and ID_odbornosti=@pom_odbor)=1)
            begin
                  insert into
provedene_zakroky(rodne_cislo,ID_lekar,ID_zakrok,datum,poznamky)
                  values
(@rodne_cislo,@ID_lekar,@ID_zakrok,@datum,@poznamky)
                  print ('Zákrok byl vytvořen.')
            end
else
      print 'Lékař nemá požadovanou odbornost!'
end
```

Příklady dotazů na přidání zákroku:

```
exec pridej_zakrok 8007112658,2,3,'2010-04-15',''
exec pridej_zakrok 8007112658,2,1,'2010-04-15',''
exec pridej_zakrok 5112172134,3,1,'2010-04-15',''
exec pridej_zakrok 5112172134,3,2,'2010-04-15',''
exec pridej_zakrok 6802276342,3,1,'2010-04-15',''
exec pridej_zakrok 6802276342,2,3,'2010-04-15',''
```

Stejný způsobem je řešena i procedura na přidání žádosti o odborné vyšetření.

4.4.3 Trigger – Aktulní datum

Tento trigger při vložení nového zubního kříže pacienta do tabulky *zubni_kriz* aktualizuje *datum_posledni_navstevy* v tabulce *pacient*.

```
create trigger aktual_datum on zubni_kriz
for insert
as
begin
      update pacient set datum_posledni_navstevy = GETDATE()
      where (rodne_cislo=(select rodne_cislo from zubni_kriz
      where ID_zubni_kriz=(select IDENT_CURRENT('zubni_kriz'))))
end
```

Příklady insertů do tabulky *zubni_kriz*:

```
insert into zubni_kriz
(rodne_cislo,ID_lekar,zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub
10,zub11,zub12,zub13,zub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,z
ub22,zub23,zub24,zub25,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum
,poznamky)
values(6802276342,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,GETDATE(),'')

insert into zubni_kriz
(rodne_cislo,ID_lekar,zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub
10,zub11,zub12,zub13,zub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,z
ub22,zub23,zub24,zub25,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum
,poznamky)
values(6802276342,3,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,2,1,1,2,1,1,1,1,2,1,
1,1,1,1,1,1,1,GETDATE(),'')
```

Příklad dotazu select, který zobrazí všechny zubní kříže pacienta s rodným číslem 680227/6342:

```
select
zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub10,zub11,zub12,zub13,z
ub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,zub22,zub23,zub24,zub25
,zub26, zub27, zub28, zub29, zub30, zub31, zub32, datum, poznamky from
zubni_kriz where rodne_cislo=6802276342
```

4.4.4 Trigger – Vyřízení žádosti

Tento trigger pracuje nad tabulkou *provedene_zakroky*. Při přidání zákroku do této tabulky se zkontroluje, zda není v tabulce *zadost_o_odborne_vysetreni* záznam se stejným rodným číslem, ID zákroku a ID lékaře. Pokud takový záznam nalezne, přidá do *splneno* v tabulce *zadost_o_odborne_vysetreni* ID právě vytvořeného zákroku.

```
create trigger vyrizeni_zadosti on provedene_zakroky
for insert
as
begin
      update zadost_o_odborne_vysetreni set splneno = (select
      IDENT_CURRENT('provedene_zakroky'))
      where ((rodne_cislo=(select rodne_cislo from provedene_zakroky
      where ID_proved_zakrok=(select
      IDENT_CURRENT('provedene_zakroky')))) and
      (ID_zakrok=(select ID_zakrok from provedene_zakroky
      where ID_proved_zakrok=(select
      IDENT_CURRENT('provedene_zakroky')))) and
      (pozadovany_lekar=(select ID_lekar from provedene_zakroky
      where ID_proved_zakrok=(select
      IDENT_CURRENT('provedene_zakroky')))))
end
```

4.4.5 Pohled – Celkový výkaz

Tento pohled slouží k tisku všech provedených zákroků včetně přiřazení výše úhrad od zdravotní pojišťovny, které jsou uvedeny v tabulce *naklady_zakroku*.

```
Create view celkovy_vykaz as
  select
  z.datum,z.rodne_cislo,p.prijmeni,p.jmeno,z.ID_zakrok,n.uhrada_ZP
  From provedene_zakroky z
     left JOIN pacient p on z.rodne_cislo = p.rodne_cislo
     left JOIN naklady_zakroku n on z.ID_zakrok = n.ID_zakrok
  where p.kod_pojistovny=n.kod_pojistovny

select * from celkovy_vykaz
```

4.5 Přínosy pro stomatologickou kliniku

Vzhledem k tomu, že cílem bakalářské práce bylo pouze navrhnout databázi a ne celý informační systém, nelze identifikovat žádné měřitelné přínosy.

I neměřitelné přínosy je těžké popsat, protože se skutečně projeví až po vytvoření aplikace, která bude databázi obsluhovat. Zde se nachází alespoň identifikace přínosů, které vyplývají z použití elektronické (v tomto případě relační) databáze:

- Zrychlení a zefektivnění práce lékaře a sestry.
- Snadná změna osobních údajů pacienta.
- Vytvoření nového zubního kříže pacienta na základě toho předchozího.
- Snadné zobrazení a tisku dokumentace (výkazů) pro zdravotní pojišťovny.
- Přenos rentgenových snímků v elektronické podobě.
- Zrychlení komunikace mezi lékaři při podávání žádostní o odborné vyšetření a rentgenové snímky.
- Kontrola odbornostních způsobilostí lékařů.
- Kontrola doplatků za zákroky nehrazených zdravotní pojišťovnou.
- Možnost zobrazení volných termínů lékařů na případné internetové stránce.
- Možnost zasílání automatických upomínek pomocí e-mailů.
- Možnost získávání souhrnných informací z databáze.

5 Závěr

Cílem mé bakalářské práce bylo navrhnout databázi pro stomatologickou kliniku. Databáze měla obsahovat evidenci jednotlivých lékařů a jejich pacientů a mělo být možné ukládat záznamy o vyšetření nebo zákroku v podobě zubního kříže a dekursu (lékařského zápisu). Dalším požadavkem bylo také ukládání rentgenových snímků v elektronické podobě a tisknutí sestav, které by sloužily jako výkaz pro zdravotní pojišťovny.

Na základě popisu současné situace a analýzy problému byla navrhnuta databáze, která splňuje všechny nároky a požadavky kladené na budoucí informační systém.

Tato bakalářská práce splnila svůj cíl a navržená databáze by měla zjednodušit a zefektivnit práci v zubní ordinaci i celé klinice.

Dalším důležitým krokem, který je nezbytný k příjemnému a efektivnímu využívání databáze, by bylo navržení uživatelského prostředí a aplikace, která by pracovala nad touto databází. Také by bylo vhodné navrhnout internetové stránky, kde by bylo online zobrazení volných termínů lékaře.

6 Seznam použitých zdrojů

6.1 Monografické zdroje

[1] CONOLLY, T., BEGG, C., HOLOWCZAK, R. *Mistrovství – databáze: Profesionální průvodce tvorbou efektivních databází.* 1. vydání. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.

[2] DOSEDĚL, T. *Počítačová bezpečnost a ochrana dat*. 1. vydání. Brno: Computer Press, 2004. 190s. ISBN 80-251-0106-1.

[3] GROFF, J., WEINBERG, P. *SQL Kompletní průvodce*. 1. vydání. Brno: Computer Press, 2005. 936 s. ISBN 80-251-0369-2.

[4] HOTEK, M. *Microsoft SQL Server 2008: krok za krokem.* 1.vydání. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6.

[5] HOULETTE, F. *SQL: Příručka programátora.* 1. vydání. Praha: SoftPress, 2001. 382 s. ISBN 80-86497-14-3.

[6] KOCH, M., NEUWIRTH, B. *Datové a funkční modelování.* 3. přepracované vydání. Brno: CERM, 2008. 121 s. ISBN 978-80-214-3731-9.

[7] KOTLER, P. *Marketing management*. 1. vydání. Praha: Grada, 2007. 788 s. ISBN 978-80-247-1359-5.

[8] KRIEGEL, A., TRUKHNOV, B. *SQL bible.* 1. vydání. Indianapolis: Wiley Pub., 2003. 831 s. ISBN 0-7645-2584-0.

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

[10] PUŽMANOVÁ, R. *Moderní komunikační sítě od A do Z*. 2. aktualizované vydání. Brno: Computer Press, 2006. 430 s. ISBN 80-251-1278-0.

[11] ŘEPA, V. *Podnikové procesy.* 2.rozšírené vydání. Praha: Grada, 2007. 281 s. ISBN 978-80-247-2252-8.

[12] VORÍŠEK, J. *Principy a modely řízení podnikové informatiky.* 1. vydání. Praha: Oeconomica, 2008. 446 s. ISBN 978-80-245-1440-6.

6.2 Internetové zdroje

[13] SKŘIVAN J., *Databáze a jazyk SQL*. [online]. 2000 [2009-10-12]. Dostupné na: <http://interval.cz/clanky/databaze-a-jazyk-sql/>.

[14] *Denram Graphics & Printing.* [online]. 2006 [2010-04-08]. Dostupné na: <http://denram.com/>.

[15] *DentimMax.* [online]. 2007 [2010-04-17]. Dostupné na: <http://dentimax.com>.

[16] *Dentist+*. [online]. [2010-04-15]. Dostupné na: <http://dentist.cz>.

[17] *Jazyk SQL*. [online]. 2006 [2010-04-09]. Dostupné na: <http://www.epos.cz/obis4/techinfo/howto/html/node4.html>.

[18] *Tomšů software s.r.o.* [online]. 2008 [2010-04-16]. Dostupné na: <http://tomsusoftware.cz>.

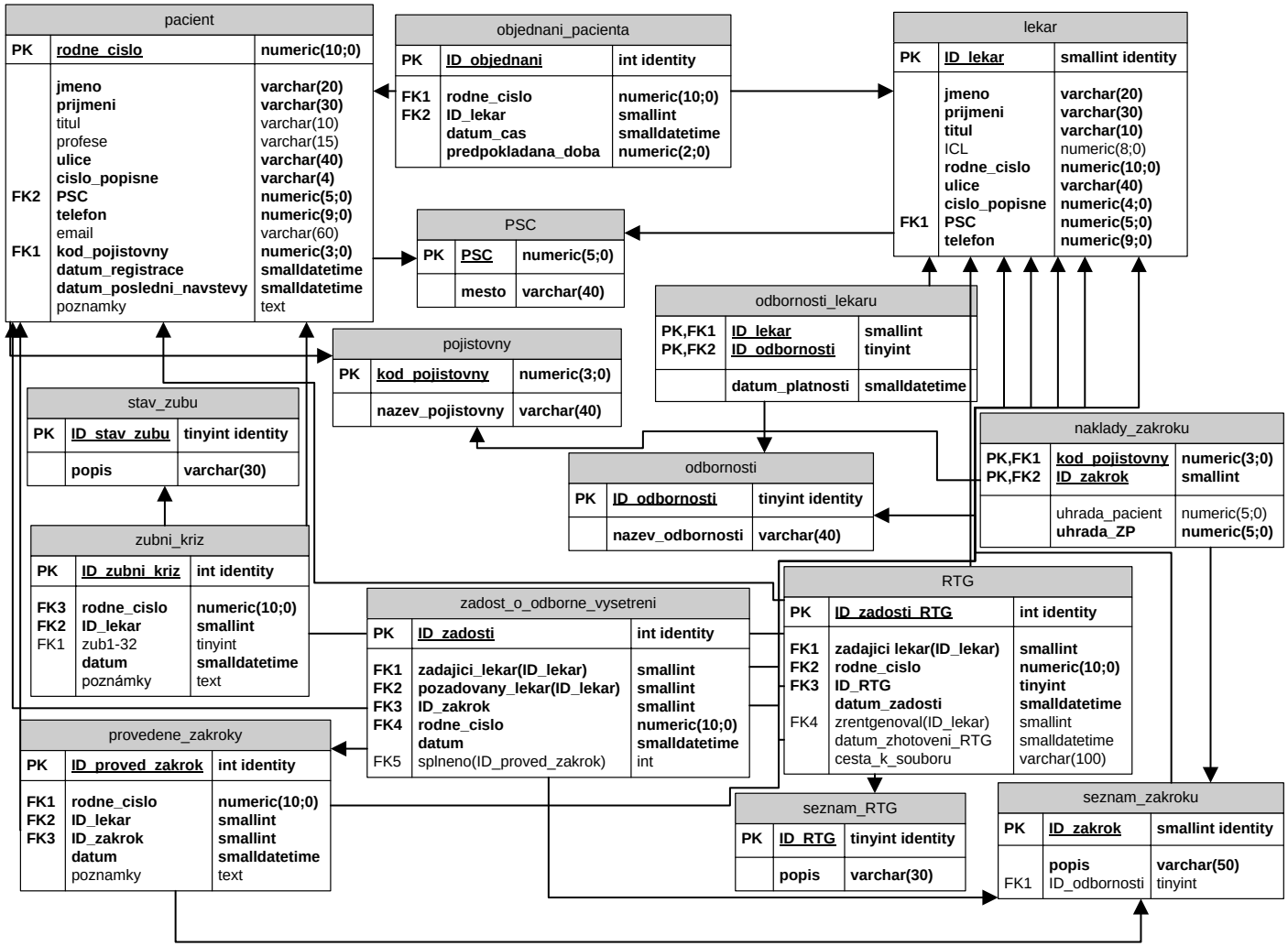
7 Seznam obrázků a tabulek

7.1 Seznam obrázků

7.2 Seznam tabulek

8 Seznam příloh

Příloha 1 - Fyzický návrh databáze



Příloha 2 - Datový slovník

Tabulka	Atributy	Datový typ	Délka	PK	Cizí klíč - tabulka (atribut)	Not NULL	Poznámky
lekar	**ID_lekar**	smallint		X		X	identity (1,1)
	jmeno	varchar	20			X	
	prijmeni	varchar	30			X	
	titul	varchar	10			X	
	ICL	numeric	8;0				
	rodne_cislo	numeric	10;0			X	
	ulice	varchar	40			X	
	cislo_popisne	numeric	4;0			X	
	PSC	numeric	5;0		PSC (PSC)	X	
	telefon	numeric	9;0			X	
naklady_zakroku	***kod_pojistovny***	numeric	3;0	X	pojistovny (kod_pojistovny)		
	ID_zakrok	smallint		X	seznam_zakroku (ID_zakrok)	X	
	uhrada_pacient	numeric	5;0				
	uhrada_ZP	numeric	8;0			X	
objednani_pacienta	**ID_objednani**	int		X		X	identity (1,1)
	rodne_cislo	numeric	10;0		pacient (rodne_cislo)	X	
	ID_lekar	smallint			lekar (ID_lekar)	X	
	datum_cas	smalldatetime				X	
	predpokladana_doba	numeric	2;0			X	
odbornosti	**ID_odbornosti**	tinyint		X		X	identity (1,1)
	nazev_odbornosti	varchar	40			X	

Tabulka	Atributy	Datový typ	Délka	PK	Cizí klíč - tabulka (atribut)	Not NULL	Poznámky
odbornosti_lekaru	***ID_lekar***	smallint		X	lekar (ID_lekar)	X	
	ID_odbornosti	tinyint		X	odbornosti (ID_odbornosti)	X	
	datum_platnosti	smalldatetime				X	
pacient	**rodne_cislo**	numeric	10;0	X		X	
	jmeno	varchar	20			X	
	prijmeni	varchar	30			X	
	titul	varchar	10				
	profese	varchar	15				
	ulice	varchar	40			X	
	cislo_popisne	varchar	4			X	
	PSC	numeric	5;0		PSC (PSC)	X	
	telefon	numeric	9;0			X	
	email	varchar	60				
	kod_pojistovny	numeric	3;0		pojistovny (kod_pojistovny)	X	
	datum_registrace	smalldatetime				X	
	datum_posledni_navstevy	smalldatetime				X	
	poznamky	text					
pojistovny	**kod_pojistovny**	numeric	3;0	X		X	
	nazev_pojistovny	varchar	40			X	
provedene_zakroky	**ID_proved_zakroku**	int		X		X	identity (1,1)
	rodne_cislo	numeric	10;0		pacient (rodne_cislo)	X	
	ID_lekar	smallint			lekar (ID_lekar)	X	
	ID_zakrok	smallint			seznam_zakroku (ID_zakrok)	X	
	datum	smalldatetime				X	
	poznamky	text					

Tabulka	Atributy	Datový typ	Délka	PK	Cizí klíč - tabulka (atribut)	Not NULL	Poznámky
PSC	**PSC**	numeric	5;0	X		X	
	město	varchar	40			X	
RTG	**ID_zadosti_RTG**	int		X		X	identity (1,1)
	zadajici_lekar	smallint			lekar (ID_lekar)	X	
	rodne_cislo	numeric	10;0		pacient (rodne_cislo)	X	
	ID_RTG	tinyint			seznam_RTG (ID_RTG)	X	
	datum_zadosti	smalldatetime				X	
	zrentgenoval	smallint			lekar (ID_lekar)		
	datum_zhotoveni_RTG	smalldatetime					
	cesta_k_souboru	varchar	100				
seznam_RTG	**ID_RTG**	tinyint		X		X	identity (1,1)
	popis	varchar	30			X	
seznam_zakroku	**ID_zakrok**	smallint		X		X	identity (1,1)
	popis	varchar	50			X	
	ID_odbornosti	tinyint			odbornosti (ID_odbornosti)		
stav_zubu	**ID_stav_zubu**	tinyint		X		X	identity (1,1)
	popis	varchar	30				

Tabulka	Atributy	Datový typ	Délka	PK	Cizí klíč - tabulka (atribut)	Not NULL	Poznámky
zadost_o_ odborne_vysetreni	**ID_zadosti**	int		X		X	identity (1,1)
	zadajici_lekar	smallint			lekar (ID_lekar)	X	
	pozadovany_lekar	smallint			lekar (ID_lekar)	X	
	ID_zakrok	smallint			seznam_zakroku (ID_zakrok)	X	
	rodne_cislo	numeric			pacient (rodne_cislo)	X	
	datum	smalldatetime				X	
	splneno	int			provedene_zakroky (ID_proved_zakrok)		
zubni_kriz	**ID_zubni_kriz**	int		X		X	identity (1,1)
	rodne_cislo	numeric	10;0		pacient (rodne_cislo)	X	
	ID_lekar	smallint			lekar (ID_lekar)	X	
	zub1-32	tinyint			stav_zubu (ID_stav_zubu)		
	datum	smalldatetime				X	
	poznamky	text					

Příloha 3 - Zdrojový kód

```
create table pojistovny
(
	kod_pojistovny numeric (3) primary key NOT NULL,
	pojistovna varchar (40) NOT NULL
)

create table PSC
(
	PSC numeric (5) primary key NOT NULL,
	mesto varchar (40) NOT NULL
)

create table seznam_RTG
(
	ID_RTG tinyint identity (1,1) primary key NOT NULL,
	popis varchar (30) NOT NULL
)

create table stav_zubu
(
	ID_stav_zubu tinyint identity (1,1) primary key NOT NULL,
	popis varchar (30) NOT NULL
)

create table odbornosti
(
	ID_odbornosti tinyint identity (1,1) primary key NOT NULL,
	popis varchar (40) NOT NULL
)

Create table pacient
(
	rodne_cislo numeric (10) primary key NOT NULL,
	jmeno varchar (20) NOT NULL,
	prijmeni varchar (30) NOT NULL,
	titul varchar (10),
	profese varchar (15),
	ulice varchar (40) NOT NULL,
	cislo_popisne numeric (4) NOT NULL,
	PSC numeric (5) foreign key references PSC (PSC) NOT NULL,
	telefon numeric (9) NOT NULL,
	email varchar (60),
	kod_pojistovny numeric (3) foreign key references Pojistovny
	(Kod_pojistovny) NOT NULL,
	datum_registrace smalldatetime NOT NULL,
	datum_posledni_navstevy smalldatetime NOT NULL,
	poznamky text
)
```

```
Create table lekar
(
      ID_lekar smallint identity (1,1) primary key NOT NULL,
      jmeno varchar (20) NOT NULL,
      prijmeni varchar (30) NOT NULL,
      titul varchar (10) NOT NULL,
      ICL numeric (8),
      rodne_cislo numeric (10) NOT NULL,
      ulice varchar (40) NOT NULL,
      cislo_popisne numeric (4) NOT NULL,
      PSC numeric (5) foreign key references PSC (PSC) NOT NULL,
      telefon numeric (9) NOT NULL
)

Create table objednani_pacienta
(
      ID_objednani int identity (1,1) primary key NOT NULL,
      rodne_cislo numeric (10) foreign key references pacient
      (rodne_cislo) NOT NULL,
      ID_lekar smallint foreign key references lekar (ID_lekar) NOT
      NULL,
      datum_cas smalldatetime NOT NULL,
      predpokladana_doba numeric (2) NOT NULL
)

Create table odbornosti_lekaru
(
      ID_lekar smallint foreign key references lekar (ID_lekar) NOT
      NULL,
      ID_odbornosti tinyint foreign key references odbornosti
      (ID_odbornosti) NOT NULL,
      datum_platnosti smalldatetime NOT NULL,
      Primary key (ID_lekar, ID_odbornosti)
)

Create table seznam_zakroku
(
      ID_zakrok smallint identity (1,1) primary key NOT NULL,
      popis varchar(50) NOT NULL,
      ID_odbornosti tinyint foreign key references odbornosti
      (ID_odbornosti)
)

Create table naklady_zakroku
(
      kod_pojistovny numeric (3) foreign key references pojistovny
      (kod_pojistovny) NOT NULL,
      ID_zakrok smallint foreign key references seznam_zakroku
      (ID_zakrok) NOT NULL,
      uhrada_pacient numeric (5),
      uhrada_ZP numeric (5) NOT NULL
)
```

```
Create table zubni_kriz
(
      ID_zubni_kriz int identity (1,1) primary key NOT NULL,
      rodne_cislo numeric (10) foreign key references pacient
      (rodne_cislo) NOT NULL,
      ID_lekar smallint foreign key references lekar (ID_lekar) NOT
      NULL,
      zub1 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub2 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub3 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub4 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub5 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub6 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub7 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub8 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub9 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub10 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub11 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub12 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub13 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub14 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub15 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub16 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub17 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub18 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub19 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub20 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub21 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub22 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub23 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub24 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub25 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub26 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub27 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub28 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub29 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub30 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub31 tinyint foreign key references stav_zubu (ID_stav_zubu),
      zub32 tinyint foreign key references stav_zubu (ID_stav_zubu),
      datum smalldatetime NOT NULL,
      poznamky text
)

Create table provedene_zakroky
(
      ID_proved_zakrok int identity (1,1) primary key NOT NULL,
      rodne_cislo numeric (10) foreign key references pacient
      (rodne_cislo) NOT NULL,
      ID_lekar smallint foreign key references lekar (ID_lekar) NOT
      NULL,
      ID_zakrok smallint foreign key references seznam_zakroku
      (ID_zakrok) NOT NULL,
      datum smalldatetime NOT NULL,
      poznamky text
)
```

```
Create table zadost_o_odborne_vysetreni
(
	ID_zadosti int identity (1,1) primary key NOT NULL,
	zadajici_lekar smallint foreign key references lekar (ID_lekar)
	NOT NULL,
	pozadovany_lekar smallint foreign key references lekar
	(ID_lekar) NOT NULL,
	ID_zakrok smallint foreign key references seznam_zakroku
	(ID_zakrok) NOT NULL,
	rodne_cislo numeric (10) foreign key references pacient
	(rodne_cislo) NOT NULL,
	datum smalldatetime NOT NULL,
	splneno int foreign key references provedene_zakroky
	(ID_proved_zakrok)
)

Create table RTG
(
	ID_zadosti_RTG int identity (1,1) primary key NOT NULL,
	zadajici_lekar smallint foreign key references lekar (ID_lekar)
	NOT NULL,
	rodne_cislo numeric (10) foreign key references pacient
	(rodne_cislo) NOT NULL,
	ID_RTG tinyint foreign key references seznam_RTG (ID_RTG) NOT
	NULL,
	datum_zadosti smalldatetime NOT NULL,
	zrentgenoval smallint foreign key references lekar (ID_lekar),
	datum_zhotoveni_RTG smalldatetime,
	cesta_k_souboru varchar (100)
)
```

```
insert into pojistovny (kod_pojistovny, pojistovna)
  values (111,'Všeobecná zdravotní pojišťovna')

insert into pojistovny (Kod_pojistovny, Pojistovna)
  values (201,'Vojenská zdravotní pojišťovna')

insert into pojistovny (Kod_pojistovny, Pojistovna)
  values (211,'Zdravotní pojišťovna ministerstva vnitra')


insert into PSC (PSC,mesto) values (60200, 'Brno')
insert into PSC (PSC,mesto) values (53002, 'Pardubice')
insert into PSC (PSC,mesto) values (11004, 'Praha')

insert into seznam_RTG (popis) values ('Celkový rentgen')
insert into seznam_RTG (popis) values ('Horní čelist')
insert into seznam_RTG (popis) values ('Dolní čelist')

insert into stav_zubu (popis) values ('Zdravý')
insert into stav_zubu (popis) values ('Plomba')
insert into stav_zubu (popis) values ('Korunka')

insert into odbornosti (popis) values ('Ortodontista')
insert into odbornosti (popis) values ('Čelistní ortoped')
insert into odbornosti (popis) values ('Stomatologické LSPP')


insert into RTG(zadajici_lekar,rodne_cislo,ID_RTG,datum_zadosti)
values (2,6802276342,1,GETDATE())

insert into RTG(zadajici_lekar,rodne_cislo,ID_RTG,datum_zadosti)
values (3,8007112658,2,GETDATE())



Insert into
pacient(rodne_cislo,jmeno,prijmeni,titul,profese,ulice,cislo_popisne,P
SC,telefon,email,kod_pojistovny,datum_registrace,datum_posledni_navste
vy, poznamky)
  values(6802276342,'Radek', 'Nový', 'Ing.', 'zedník' ,
'Brněnská',1,60200,737304618, 'novy.r@seznam.cz',111,
GETDATE(),GETDATE(),'Alergik, cukrovka')

Insert into
pacient(rodne_cislo,jmeno,prijmeni,profese,ulice,cislo_popisne,PSC,tel
efon,email,kod_pojistovny,datum_registrace,datum_posledni_navstevy)
  values(8007112658,'Michal', 'Hrdý', 'právník' ,
'Letecká',234,53002,721302334, 'hrdy.michal@seznam.cz',201,
GETDATE(),GETDATE())

Insert into
pacient(rodne_cislo,jmeno,prijmeni,profese,ulice,cislo_popisne,PSC,tel
efon,kod_pojistovny,datum_registrace,datum_posledni_navstevy)
  values(5112172134,'Bohumil', 'Krásný', 'učetní' ,
'Benešova',15,60200,775267887,111, GETDATE(),GETDATE())


Insert into lekar (jmeno, prijmeni, titul, ICL, rodne_cislo, ulice,
cislo_popisne, PSC, telefon)
```

```
  values
('Marek','Kroupa','MUDr.',25438675,7012129876,'Dlouhá',1,60200,6023445
66)

Insert into lekar (jmeno, prijmeni, titul, rodne_cislo, ulice,
cislo_popisne, PSC, telefon)
  values
('Pilná','Lucie','MUDr.',7852102111,'Kratká',1111,60200,608909392)


Insert into odbornosti_lekaru (ID_lekar,ID_odbornosti,datum_platnosti)
  values(2,1, '2011-10-04')
Insert into odbornosti_lekaru (ID_lekar,ID_odbornosti,datum_platnosti)
    values(2,2, '2008-10-04')
Insert into odbornosti_lekaru (ID_lekar,ID_odbornosti,datum_platnosti)
    values(3,2, '2012-07-11' )
Insert into odbornosti_lekaru (ID_lekar,ID_odbornosti,datum_platnosti)
    values(3,3, '2009-07-11' )

Insert into seznam_zakroku (popis,ID_odbornosti)
  values('Preventivní prohlídka', NULL)
Insert into seznam_zakroku (popis,ID_odbornosti)
  values('Extrakce zubu', NULL)
Insert into seznam_zakroku (popis,ID_odbornosti)
  values('Vstupní ortodontická konzultace', 1)



insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(111,1,30,250)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(111,2,0,90)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(111,3,500,500)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(201,1,30,200)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(201,2,0,130)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(201,3,300,700)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(211,1,30,230)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(211,2,0,50)
insert into
naklady_zakroku(kod_pojistovny,ID_zakrok,uhrada_pacient,uhrada_ZP)
values(211,3,400,600)
```

```
create procedure prover_odbornosti
      @dat1 smalldatetime
as
begin
      declare
      @id_odbor tinyint,
      @id_lek smallint,
      @dat2 smalldatetime

declare cur_odbor cursor for
      select o.ID_odbornosti, o.ID_lekar, o.datum_platnosti
      from odbornosti_lekaru o
open cur_odbor
      fetch next from cur_odbor into @id_odbor, @id_lek, @dat2
while @@fetch_status=0
      begin
if (@dat2<@dat1)
      begin
delete from odbornosti_lekaru where (ID_odbornosti=@id_odbor) and
(ID_lekar=@id_lek)

      end
      fetch next from cur_odbor into @id_odbor, @id_lek, @dat2
      end
close cur_odbor
deallocate cur_odbor
end


exec prover_odbornosti '2010-04-15'

select * from odbornosti_lekaru
```

```
create procedure pridej_zakrok
  @rodne_cislo numeric(10),
  @ID_lekar smallint,
  @ID_zakrok smallint,
  @datum smalldatetime,
  @poznamky text

as
begin
      declare
      @pom_odbor tinyint
      set @pom_odbor = (select ID_odbornosti from seznam_zakroku where
ID_zakrok=@ID_zakrok)

if ((@pom_odbor is NULL) or (select count(ID_odbornosti) from
odbornosti_lekaru
      where ID_lekar=@ID_lekar and ID_odbornosti=@pom_odbor)=1)
            begin
                  insert into
provedene_zakroky(rodne_cislo,ID_lekar,ID_zakrok,datum,poznamky)
                  values
(@rodne_cislo,@ID_lekar,@ID_zakrok,@datum,@poznamky)
                  print ('Zákrok byl vytvořen.')
            end
else
      print 'Lékař nemá požadovanou odbornost!'
end


exec pridej_zakrok 8007112658,2,3,'2010-04-15',''
exec pridej_zakrok 8007112658,2,1,'2010-04-15',''
exec pridej_zakrok 5112172134,3,1,'2010-04-15',''
exec pridej_zakrok 5112172134,3,2,'2010-04-15',''
exec pridej_zakrok 6802276342,3,1,'2010-04-15',''
exec pridej_zakrok 6802276342,2,3,'2010-04-15',''

select * from provedene_zakroky
```

```
create procedure pridej_zadost
  @zadajici_lekar smallint,
  @pozadovany_lekar smallint,
  @ID_zakrok smallint,
  @rodne_cislo numeric(10)

as
begin
      declare
      @pom_odbor tinyint
      set @pom_odbor = (select ID_odbornosti from seznam_zakroku where
ID_zakrok=@ID_zakrok)

if ((@pom_odbor is NULL) or (select count(ID_odbornosti) from
odbornosti_lekaru
      where ID_lekar=@pozadovany_lekar and
ID_odbornosti=@pom_odbor)=1)
            begin
                  insert into
zadost_o_odborne_vysetreni(zadajici_lekar,pozadovany_lekar,ID_zakrok,r
odne_cislo,datum)
                  values
(@zadajici_lekar,@pozadovany_lekar,@ID_zakrok,@rodne_cislo,GETDATE())
                  print ('Žádost byla vytvořena.')
            end
else
      print 'Lékař nemá požadovanou odbornost!'
end


exec pridej_zadost 3,2,3,8007112658
exec pridej_zadost 3,2,3,5112172134

select * from zadost_o_odborne_vysetreni



select zadajici_lekar,ID_zakrok,rodne_cislo,datum from
zadost_o_odborne_vysetreni where splneno is NULL and
pozadovany_lekar=2
```

```
create trigger aktual_datum on zubni_kriz
for insert
as
begin
      update pacient set datum_posledni_navstevy = GETDATE()
      where (rodne_cislo=(select rodne_cislo from zubni_kriz
      where ID_zubni_kriz=(select IDENT_CURRENT('zubni_kriz'))))
end

insert into zubni_kriz
(rodne_cislo,ID_lekar,zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub
10,zub11,zub12,zub13,zub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,z
ub22,zub23,zub24,zub25,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum
,poznamky)
values(6802276342,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,GETDATE(),'')

insert into zubni_kriz
(rodne_cislo,ID_lekar,zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub
10,zub11,zub12,zub13,zub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,z
ub22,zub23,zub24,zub25,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum
,poznamky)
values(6802276342,3,1,1,1,1,1,1,1,1,1,1,2,1,1,1,2,1,1,2,1,1,1,1,2,1,
1,1,1,1,1,1,1,GETDATE(),'')

insert into zubni_kriz
(rodne_cislo,ID_lekar,zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub
10,zub11,zub12,zub13,zub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,z
ub22,zub23,zub24,zub25,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum
,poznamky)
values(6802276342,3,1,1,1,1,1,1,2,1,1,1,1,2,1,1,1,2,1,1,2,1,1,3,3,2,1,
1,1,2,1,1,1,1,GETDATE(),'')

insert into zubni_kriz
(rodne_cislo,ID_lekar,zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub
10,zub11,zub12,zub13,zub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,z
ub22,zub23,zub24,zub25,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum
,poznamky)
values(8007112658,2,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,
1,1,1,1,1,1,1,GETDATE(),'')


select * from pacient
select * from zubni_kriz



select
zub1,zub2,zub3,zub4,zub5,zub6,zub7,zub8,zub9,zub10,zub11,zub12,zub13,z
ub14,zub15,zub16,zub17,zub18,zub19,zub20,zub21,zub22,zub23,zub24,zub25
,zub26,zub27,zub28,zub29,zub30,zub31,zub32,datum,poznamky from
zubni_kriz where rodne_cislo=6802276342
```

```
create trigger vyrizeni_zadosti on provedene_zakroky
for insert
as
begin
      update zadost_o_odborne_vysetreni set splneno = (select
IDENT_CURRENT('provedene_zakroky'))
      where ((rodne_cislo=(select rodne_cislo from provedene_zakroky
      where ID_proved_zakrok=(select
IDENT_CURRENT('provedene_zakroky')))) and
      (ID_zakrok=(select ID_zakrok from provedene_zakroky
      where ID_proved_zakrok=(select
IDENT_CURRENT('provedene_zakroky')))) and
      (pozadovany_lekar=(select ID_lekar from provedene_zakroky
      where ID_proved_zakrok=(select
IDENT_CURRENT('provedene_zakroky')))))
end

exec pridej_zakrok 5112172134,2,3,'2010-04-22','Vyřízení žádosti.'
select * from provedene_zakroky
select * from zadost_o_odborne_vysetreni
```

```
Create view celkovy_vykaz as
  select
  z.datum,z.rodne_cislo,p.prijmeni,p.jmeno,z.ID_zakrok,n.uhrada_ZP
  From provedene_zakroky z
     left JOIN pacient p on z.rodne_cislo = p.rodne_cislo
     left JOIN naklady_zakroku n on z.ID_zakrok = n.ID_zakrok
  where p.kod_pojistovny=n.kod_pojistovny

select * from celkovy_vykaz


select
  z.datum,z.rodne_cislo,p.prijmeni,p.jmeno,z.ID_zakrok,n.uhrada_ZP
  From provedene_zakroky z
     left JOIN pacient p on z.rodne_cislo = p.rodne_cislo
     left JOIN naklady_zakroku n on z.ID_zakrok = n.ID_zakrok
  where p.kod_pojistovny=n.kod_pojistovny and p.kod_pojistovny=111 and
z.ID_lekar=3
```