Obrázok, na ktorom je text

Automaticky generovaný popis

SEMESTRÁLNA PRÁCA

**Pokročilé databázové systémy**

**Nemocničný informačný systém**

**Vypracovali:**

Marek Šútora   (5ZIB11)         **Akademický rok:** 2022/2023

Samuel Slivovský (5ZIB11)         **Študijný program:**

Iveta Šinálová (5ZIB11)         Biomedicínska informatika

Tomáš Štulrajter (5ZIB11)

Tomáš Labát (5ZIB11)

Obsah

[Popis semestrálnej práce 3](#_Toc121076644)

[Dátový model 3](#_Toc121076645)

[PL/SQL 6](#_Toc121076646)

[Procedúry 6](#_Toc121076647)

[Funkcie 7](#_Toc121076648)

[Výstupy 9](#_Toc121076649)

[Štatistické výstupy 9](#_Toc121076650)

[Ostatné výstupy 10](#_Toc121076651)

[XML Report 11](#_Toc121076652)

[Analýza výkonnosti selectu vzhľadom na indexy 12](#_Toc121076653)

[Prístup k dátam na vzdialenom serveri 17](#_Toc121076654)

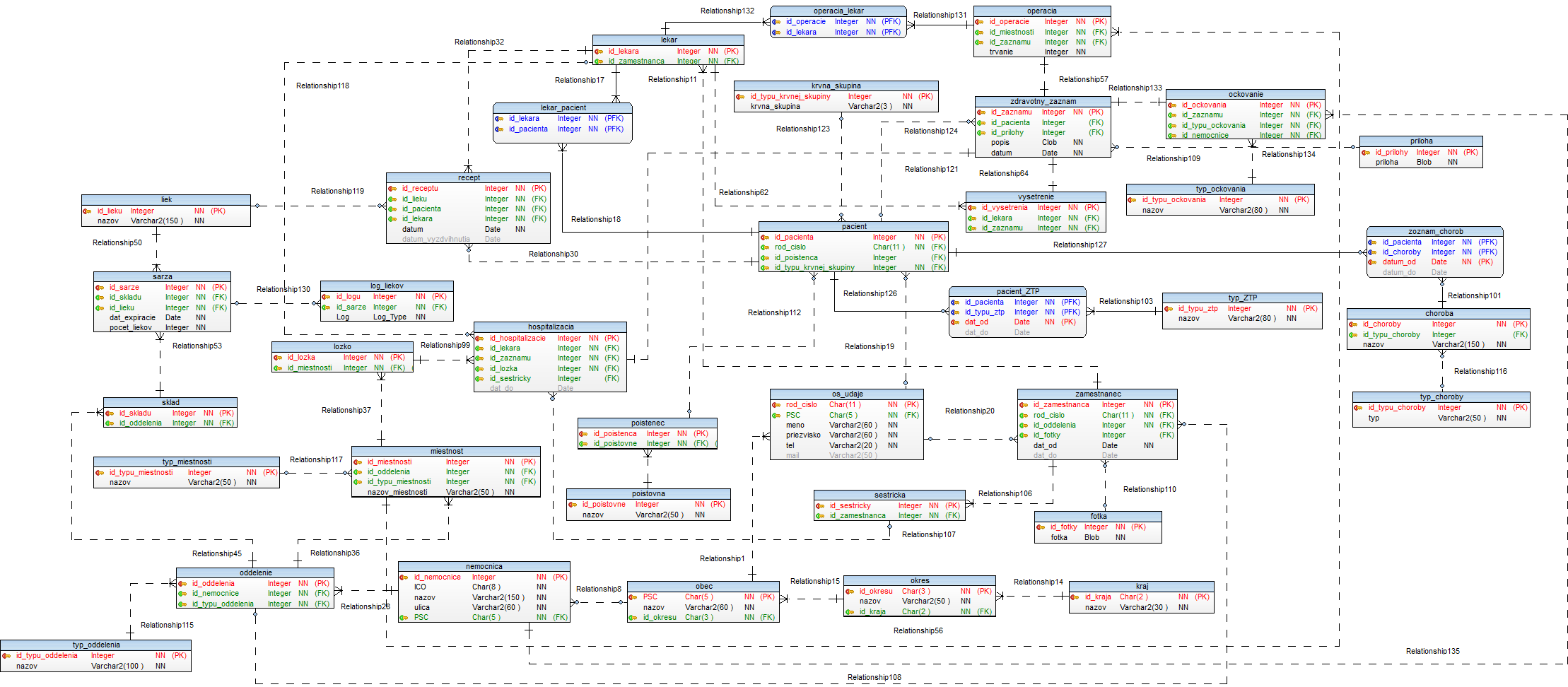
[Zoznam obrázkov 18](#_Toc121076655)

[Zoznam tabuliek 18](#_Toc121076656)

# Popis semestrálnej práce

Témou našej semestrálnej práce je nemocničný informačný systém spolu s grafickým rozhraním pre zamestnancov. Grafické rozhranie sme vytvorili pomocou frontendového javascript frameworku React a backendovú časť sme vytvorili pomocou frameworku Node.js. Súbory týkajúce sa tejto semestralnej práce sa v našom projekte nachádzajú v priečinku SemestrálnaPráca\_PDS, kde sú rozdelené do priečinkov podľa toho čoho sa týkajú. Na uloženie dát pre aplikáciu sme využili školský databázový server obelix, ktorý beží na databázovom systéme Oracle. Na testovanie boli v práce využité dáta, ktoré sa nachádzajú v priečinku Testovacie\_data.

## Dátový model



Obrázok Dátový model

Dátový model bol vytvorený v programe Toad Data Modeller a nachádza sa v priečinku Toad\_Model.

### os\_udaje

Tabuľka osobných údajov pacientov a zamestnancov nemocnice.

### zamestnanec

Tabuľka všetkých zamestnancov nemocnice, spolu s dátumom nástupu a odchodu. Každý zamestnanec má tiež pridelené oddelenie, na ktorom pracuje.

### fotka

Tabuľka, ktorá v sebe uchováva informáciu o fotke zamestnanca. Fotka je vo formáte BLOB.

### sestricka

Tabuľka, ktorá reprezentuje sestričku.

### pacient

Tabuľka reprezentujúca konkrétneho pacienta.

### pacient\_ZTP

Tabuľka ktorá určuje typ ZŤP konkrétneho pacienta a od kedy do kedy ho má.

### typ\_ZTP

Uchováva v sebe všetky typy ZŤP.

### zoznam\_chorob

Tabuľka, ktorá určuje zoznam chorôb, konkrétneho pacienta. Ku každej chorobe je aj dátum od kedy do kedy danú chorobu mal.

### choroba

Pomocou tejto tabuľky vieme určiť, ku ktorému typu choroby patrí konkrétny názov choroby.

### typ\_choroby

Obsahuje všetky typy chorôb, ktoré sa následne priraďujú k jednotlivým chorobám.

### krvna\_skupina

Tabuľka obsahujúca všetky typy krvných skupín, ktoré sú následne priradené pacientovi.

### zdravotny\_zaznam

Tabuľka obsahuje všetky zdravotné záznamy pacienta spolu s popisom a dátumom záznamu. Zastrešuje všeobecnú informáciu o očkovaniach, vyšetreniach, operáciách a hospitalizáciách pacientov. Každý záznam môže mať aj prílohu.

### priloha

Obsahuje prílohu ku konkrétnemu záznamu.

### ockovanie

Tabuľka ktorá v sebe uchováva informácie o všetkých očkovaniach, kde je uchované aj v ktorej nemocnici sa očkovanie robilo, aký má typ a o ktorý záznam sa jedná.

### typ\_ockovania

Tabuľka, ktorá obsahuje názvy typov očkovaní, ktoré sú následne priraďované k jednotlivým očkovaniam.

### vysetrenie

Tabuľka, ktorá predstavuje vyšetrenie pacienta lekárom.

### operacia

Tabuľka, ktorá predstavuje operáciu pacienta. Obsahuje informáciu o zázname pacienta a taktiež o miestnosti a trvaní danej operácie.

### operacia\_lekar

Tabuľka, ktorá zabezpečuje vzťah M:N medzi operáciou a lekárom.

### lekar

Tabuľka, v ktorej sa nachádzajú všetci zamestnanci, ktorí sú lekármi.

### lekar\_pacient

Tabuľka, v ktorej sú uložení ku každému lekárovi jeho pacienti.

### recept

Tabuľka, ktorá predstavuje recept predpísaný lekárom pacientovi. Obsahuje taktiež informácie o lieku, dátume jeho vydania a následného vyzdvihnutia.

### liek

Tabuľka, ktorá predstavuje názvy všetkých liekov používaných v databáze.

### sarza

Tabuľka, ktorá predstavuje jednu šaržu nachádzajúcu sa v sklade. Obsahuje informáciu o lieku, jeho počte v danej šarži a taktiež o dátume exspirácia.

### log\_liekov

Tabuľka, v ktorej sa zaznamenávajú informácie o zamestnancoch, ktorí pridávali alebo odoberali lieky z danej šarže. Informácia o zamestnancovi sa uchováva v atribúte Log, ktorého typ je objekt Log\_Type.

### *Log\_Type*

Objekt, ktorý je tvorený atribútmi – id\_zamestnanca, datum a zmena\_poctu. id\_zamestnanca predstavuje id zamestnanca, ktorý manipuloval s danou šaržou, datum je dátum kedy k manipulácii došlo a zmena\_poctu je zmena v počte liekov v danej šarži. Táto zmena môže byť buď kladná (v prípade pridania lieku) alebo záporná (v prípade odobrania lieku). Obsahuje taktiež 2 metódy – na výpis a taktiež na triedenie objektu. Triedenie sa vykonáva na základe atribútu zmena\_poctu. Kód k definícií tohto objektového typu spolu s jeho funkciami sa nachádza v priečinku Objekt.

### sklad

Tabuľka, ktorá zabezpečuje priradenie skladu k danému oddeleniu.

### miestnost

Tabuľka, ktorá predstavuje miestnosť v oddelení nemocnice. Obsahuje taktiež aj názov miestnosti a id jej typu.

### typ\_miestnosti

Tabuľka, ktorá obsahuje názov typu miestnosti, ktoré sú následne prideľované k miestnostiam.

### oddelenie

Tabuľka, ktorá predstavuje oddelenie v danej nemocnici.

### typ\_oddelenia

Tabuľka, ktorá obsahuje názvy jednotlivých typov oddelení, ktoré sú následne priraďované k oddeleniu.

### lozko

Tabuľka prostredníctvom ktorej sú k jednotlivým miestnostiam priraďované lôžka.

### hospitalizacia

Tabuľka, ktorá predstavuje hospitalizáciu pacienta. Obsahuje informáciu o zázname pacienta, lôžku, na ktorom je pacient hospitalizovaný, o sestričke, ktorá sa hospitalizovaného pacienta stará a o dátume kedy bol prepustený.

### nemocnica

Tabuľka, ktorá ukladá informácie o nemocnici. Konkrétne jej IČO, názov, ulicu a PSČ.

### obec

Tabuľka, v ktorej sú k PSČ priradené názvy obcí a id okresu, do ktorého obec patrí.

### okres

Tabuľka, v ktorej sa nachádzajú názvy okresov a id krajov, do ktorých dané okresy patria.

### kraj

Tabuľka, ktorá obsahuje názvy krajov.

### poistovna

Tabuľka, v ktorej sú uložené názvy poisťovní.

### poistenec

Tabuľka, pomocou ktorej je priradený poistenec ku poisťovni.

# PL/SQL

V našej práci sme v rámci využitia PL/SQL použili procedúry a funkcie, v ktorých využívame kolekcie a recordy. Všetky tieto súbory sa nachádzajú v priečinku PL\_SQL.

## Procedúry

Procedúry využívame v našej práci na generovanie náhodných dát do niektorých tabuliek. Používame ich aj na vkladanie dát do tabuľky sarza a taktiež pri vkladaní dát typu BLOB do tabuliek fotka a príloha.

### Procedúry na generovanie operácií

Procedury, ktoré slúžia na generovanie náhodných dát do tabuliek opeacia, operacia\_lekar a zdravotny\_zaznam. Nachádzajú sa v súbore operacia\_insert.sql

#### generovanie\_operacii\_p

Procedúra, ktorá slúži na generovanie náhodných dát do tabuľky **operacia**.

#### generovanie\_zdr\_zazn\_p

Procedúra, ktorá vytvorí pre generovanú operáciu zdravotný záznam.

#### generovanie\_operacia\_lekar\_p

Procedúra ktorá slúži na generovanie dát do tabuľky **operacia\_lekar** tak, aby bol priradený lekár zo správneho oddelenia.

#### generovanie\_lekar\_pacient\_operacie\_p

Procedúra ktorá zabezpečí vytvorenie vzťahu medzi lekárom a pacientom pre tabuľku **operacia**. Pomocou for cyklu voláme pre každý riadok funkciu **existuje\_lekar\_pacient\_f**. Táto funkcia je bližšie popísaná v sekcií [Funkcie](#_Funkcie).

#### generovanie\_vysetreni\_p

Procedúra ktorá slúži na generovanie dát do tabuľky **vysetrenie**. Nachádza sa v súbore vysetrenie\_insert.sql.

#### insert\_hospitalizacie

Procedúra ktorá slúži na generovanie dát do tabuľky **hospitalizacia**. Nachádza sa v súbore hospitalizacia\_insert.sql.

#### recept\_insert

Procedúra ktorá slúži na generovanie dát do tabuľky **recept.** Nachádza sa v súbore recept\_insert.sql.

#### sarza\_insert\_proc

Procedúra ktorá slúži na vloženie dát do logovacej tabuľky **log\_liekov** po vložení dát do tabuľky **sarza**. Nachádza sa v súbore sarza\_insert\_update.sql.

#### sarza\_update\_proc

Procedúra ktorá slúži na aktualizáciu tabuľky **sarza**, taktiež vloží záznam do logovacej tabuľky **log\_liekov**. Nachádza sa v súbore sarza\_insert\_update.sql.

#### uprava\_sarze

Procedúra na vytvorenie nových náhodných záznamov do tabuľky **log\_liekov** na základe zmeny počtu liekov pre danú šaržu. Nachádza sa v súbore sarza\_insert\_update.sql.

### insert\_BLOB\_into\_table

Procedúra, slúžiaca na vkladanie BLOBov do tabuliek fotka a príloha. Prvým krokom na načítanie BLOBu do databázy je nahratie súboru na server cezar.fri.uniza.sk do zložky /media/Obelix.Bloby\_student/labat\_sp (čo predstavuje login našej semestrálne práce). Takýto súbor sa automaticky mapuje na fyzickú cestu C:\Bloby\_student\labat\_sp\ . Ďalším krokom je vytvorenie directory, ktoré predstavuje fyzickú cestu k súboru. Následne si v rámci procedúry vytvoríme BFILE, ktorý pomocou konštruktora namapujeme na vytvorený directory a názov súboru, ktorý sme vložili na server cezar. Ako ďalšie vytvoríme smerník na BLOB, pomocou konštruktora EMPTY\_BLOB(), hodnota je však stále prázdna. Súbor otvoríme, získame jeho dĺžku a načítame jeho hodnotu do premennej, ktorú následne vložíme do tabuľky.

## Funkcie

Funkcie sme v našej práci využili na generovanie XML dokumentov pre požadovanú nemocnicu, ktorá sa odovzdáva prostredníctvom parametra. Jednotlivé tieto funkcie je možné nájsť taktiež v priečinku PL\_SQL/ XML\_Funkcie. Taktiež sme si vytvorili pomocnú funkcie, ktoré využívame pri generovaní dát prostredníctvom procedúr.

### existuje\_lekar\_pacient\_f

Funkcia ktorá slúži na kontrolu či existuje vzťah medzi lekárom a pacientom.

### getHospitalizacieNemocnice\_f

Funkcia ktorá slúži na vytvorenie XML dokumentu pre hospitalizácie danej nemocnice.

### getOckovaniaNemocnice\_f

Funkcia ktorá slúži na vytvorenie XML dokumentu pre očkovania danej nemocnice.

### getOperacieNemocnice\_f

Funkcia ktorá slúži na vytvorenie XML dokumentu pre operácie danej nemocnice.

### getVysetreniaNemocnice\_f

Funkcia ktorá slúži na vytvorenie XML dokumentu pre vyšetrenia danej nemocnice.

# Výstupy

Všetky výstupy sa nachádzajú v súbore vystupy.sql, ktorý sa nachádza v priečinku Vystupy. Tieto výstupy sú tam usporiadané postupne ako sú očíslované nižšie v texte.

## Štatistické výstupy

Cieľom týchto výstupov bolo vytvoriť jednoduché zobrazenie dôležitých údajov pre jednotlivé oddelenia. Výsledok sme docielili pomocou použitia grafov a tabuliek.

### Počet pacientov oddelenia:

Vypísanie počtu všetkých pacientov ktorí sa objavili na vybranom oddelení. Informuje zamestnancov oddelenia o tom, koľko pacientov je na ich oddelení. Výstup je zobrazený ako číslo.

### Počet zamestnancov oddelenia:

Vypísanie počtu zamestnancov oddelenia ktorí pracujú alebo pracovali v danom roku. Informuje zamestnancov koľko kolegov majú/mali v danom roku. Vedia si teda pozrieť či im kolegovia pribudli, prípadne odbudli. Výstup je zobrazený ako číslo.

### Počet vykonaných hospitalizácií:

Vypísanie počtu vykonaných hospitalizácií v danom roku pre vybrané oddelenie. Informuje zamestnancov o tom, koľko rôznych hospitalizácií bolo v danom roku vykonaných na ich oddelení. Výstup je zobrazený ako číslo.

### Počet vykonaných operácií:

Vypísanie počtu vykonaných operácií v danom roku pre vybrané oddelenie. Informuje zamestnancov o tom, koľko rôznych **operácií** bolo v danom roku vykonaných na ich oddelení. Výstup je zobrazený ako číslo.

### Počet vykonaných vyšetrení:

Vypísanie počtu vykonaných vyšetrení v danom roku pre vybrané oddelenie. Informuje zamestnancov o tom, koľko rôznych **vyšetrení** bolo v danom roku vykonaných na ich oddelení. Výstup je zobrazený ako číslo.

### Podiel mužských a ženských pacientov:

Vypísanie percentuálneho podielu medzi ženami a mužmi u pacientov. Informuje zamestnancov o tom, koľko pacientov predstavuje ženskú a koľko mužskú časť. Vedia si takto zistiť majoritné pohlavie. Výstup je zobrazený ako koláčový graf.

### Rozdelenie pacientov podľa veku:

Vypísanie počtu pacientov vzhľadom na vek. Informuje zamestnancov o počte pacientov pre danú vekovú kategóriu. Zamestnanci si takto vedia zistiť s akými vekovými kategóriami sa stretávali najčastejšie. Výstup je zobrazený ako stĺpcový graf.

### Výplaty v mesiacoch:

Vypísanie súčtu výplat zamestnancov v mesiacoch pre vybrané oddelenie a rok. Informuje zamestnancov o súčte výplat vo vybranom roku rozdelených po mesiacoch. Zamestnanci sa takto vedia informovať ako veľmi sa zmenili výplaty v minulých rokoch. Výstup je zobrazený ako stĺpcový graf.

### Najlepšie zarábajúci zamestnanci:

Vypísanie piatich najlepšie zarábajúcich zamestnancov. Informuje zamestnancov o ich najlepšie zarábajúcich kolegoch v danom roku. Výstup je zobrazený ako tabuľka.

### Počet typov krvnej skupiny u pacientov:

Vypísanie počtov typov krvných skupín u pacientov na vybranom oddelení. Zamestnancov takto informuje aké krvné skupiny sú pre ich oddelenie najčastejšie. Výstup je zobrazený ako koláčový graf



Obrázok Zobrazenie štatistických výstupov

## Ostatné výstupy

Tieto výstupy sú zobrazované pomocou tabuľky. Pri niektorých týchto výstupoch je možné najprv zadať určité parametre ako napríklad počet alebo percentuálny podiel prvkov.

### Pacienti s najviac chorobami

Získa top X pacientov s najviac chorobami, kde X je zadávaný parameter.

### Pacienti s najviac operáciami

Získa top X percent pacientov s najviac operáciami.

### Pacienti s najviac hospitalizáciami

Získa top X percent pacientov s najviac hospitalizáciami.

### Najlepšie platený zamestnanci

Získa top X najlepšie platených zamestnancov pre každé oddelenie.

### Počty očkovaní podľa typu pre pacientov

Pre všetkých pacientov sa vypíše koľkokrát podstúpili očkovanie daného typu.

### Všetci zamestnanci všetkých oddelení

Vypíše všetkých zamestnancov pre každé oddelenie.

### Najčastejšie choroby roka

Získa top X najčastejších chorôb vybraného roka.

### Neobsadené lôžka na najbližší týždeň na oddelení

Vypíše všetky neobsadené lôžka na najbližší týždeň pre vybrané oddelenie.

### Najviac predpisované lieky roka

Vypíše 10% najviac predpisovaných liekov vybraného roka.

### Lieky s počtom menej ako

Lieky k oddeleniam v nemocnici, ktorých je v sklade menej ako 5.

### Menovci medzi pacientami a lekármi

Vypíše krstné meno pacienta a lekára ak sú rovnaké a navyše ich priezviská.

### Operácie podľa počtu lekárov a trvania

Vypíše informácie o operáciách, ktoré spĺňajú podmienku, že počet lekárov a trvanie je väčšie ako zadané parametre.

### Kraje podľa počtu operovaných

Vráti zoznam krajov usporiadaných podľa počtu pacientov, ktorí v nich boli operovaní.

# XML Report

Pre každú nemocnicu sme vytvorili možnosť vygenerovania XML dokumentov pre výpis všetkých svojich operácií, hospitalizácií, očkovaní alebo vyšetrení. K jednotlivým záznamom sa okrem špecifických informácií pre daný typ záznamu vypíšu aj dodatočné informácie akými sú dátum a popis. Kódy k funkciám na generovanie týchto XML dokumentov sa nachádzajú v priečinku PL\_SQL/XML\_Funkcie. Tieto XML dokumenty v našej práci ale konvertujeme pomocou funkcie getClobVal() na dátový typ CLOB aby bolo možné ľahko získať tento dokument ako textový reťazec a zobraziť ho používateľovi v jeho webovom prehliadači.

# Analýza výkonnosti selectu vzhľadom na indexy

V práci sme výkonnostne analyzovali nasledujúce príkazy:

1.)

**select sum(vek) from**

**(select extract(year from sysdate) - extract(year from datum\_narodenia) as vek**

**from (select to\_date(substr(rod\_cislo, 5, 2) || '.' || (case when substr(rod\_cislo, 3, 1) = '5' then '0' when substr(rod\_cislo, 3, 1) = '6' then '1' end)**

**|| substr(rod\_cislo, 4, 1) || '.19' || substr(rod\_cislo, 1, 2), 'DD.MM.YYYY') as datum\_narodenia**

**from os\_udaje join zamestnanec using(rod\_cislo)**

**join lekar using(id\_zamestnanca)))) /**

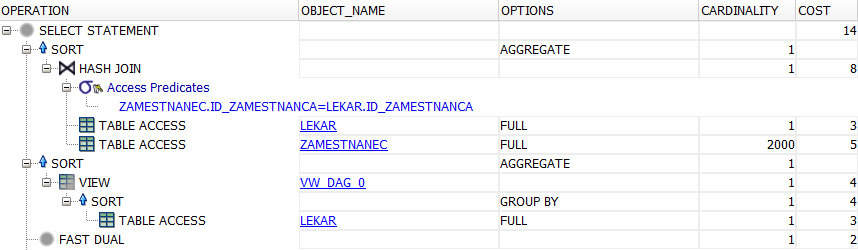
**(select count(distinct id\_zamestnanca) from lekar) as priemerny\_vek**

**from dual;**

Tento príkaz select vráti priemerný vek lekára, ktorý vypočítava pomocou vnorených selectov ako súčet vekov všetkých lekárov podelený ich počtom. Na analýzu sme využili funkciu Autotrace programu sql developer, kde sme sledovali parameter „cost”, ktorý predstavuje metriku nákladnosti vykonaného príkazu.

V prípade, že nad relevantnými stĺpcami nie je vytvorený žiadny index, odhaduje optimalizátor cost na hodnotu 14. Tabuľky sú prechádzané metódou TABLE ACCESS –FULL. Najskôr sa vykonal spodný select, kde sa kvôli použitiu klauzuly distinct vo funkcii count() vykonalo sortovanie tabuľky lekar podľa id\_zamestnanca (metóda SORT GROUP BY). Na výsledok sa použil ďalší sort, tentokrát SORT AGGREGATE, ktorý sa často volá v prípade použitia agregačnej funkcie, ktorá vracia len 1 riadok, v našom prípade count().

Následne sa vykonal vrchný select. Vzhľadom na neprítomnosť akýchkoľvek indexov sa na prístup k tabuľkám použila výlučne metóda TABLE ACCESS FULL. Na joinovanie sa použila metóda HASH JOIN. Môžeme si všimnúť, že na tabuľku os\_udaje sa nikdy nepristupovalo, nakoľko z nej prakticky nezískavame žiadne dáta, pretože údaje o rodných číslach je možné získať z tabuľky zamestnanec. Plán vykonania v základnom stave vyzerá takto:



Obrázok Plán vykonania 1

Následne sme vykonali nasledujúce optimalizácie:

Pridali sme indexy nad PK tabuliek, ktoré select používa (*os\_udaje* (rod\_cislo)), *zamestnanec* (id\_zamestnanca, *lekar*(id\_lekara)), ktoré sa štandardne vytvárajú automaticky pri vzniku tabuliek. Tento základný krok znížil cost na hodnotu 9. To nezmenilo execution plan v dolnom selecte, ale v hornom nastali nasledujúce zmeny:

Obrázok, na ktorom je text

Automaticky generovaný popisSpojenie tabuliek lekar a zamestnanec sa zmenilo na NESTED LOOPS – vonkajšou tabuľkou bol lekar (prístup cez TABLE ACCESS FULL), vnútornú tabuľku predstavoval zamestnanec, ku ktorému bol rýchly prístup kvôli indexu nad jeho PK (id\_zamestnanca) (metóda ACCESS BY INDEX ROWID, index sa prechádzal metódou INDEX UNIQUE SCAN, pretože PK má vlastnosť unikátnosti).

Obrázok 4 Plán vykonania 2

Pridali sme indexy nad FK tabuliek *zamestnanec* (rod\_cislo) a *lekar* (id\_zamestnanca), čím sme opäť výrazne znížili cost, a to na hodnotu 4. Nastali nasledujúce zmeny:

Nastala zmena prístupu k tabuľke lekar pri dolnom aj hornom selecte, kde sa namiesto TABLE ACCESS FULL použila metóda INDEX FULL SCAN, pretože prechádzanie indexu je rýchlejšie ako prechádzanie tabuľky.

Pridávaním ďalších indexov, ani funkčných by už výkon testovaného príkazu nevylepšilo. Výsledný plán vykonania je nasledujúci:

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Plán vykonania 3

2.)

**select nazov\_lieku, pocet\_predpisani, poradie from**

**(select l.nazov as nazov\_lieku, count(\*) as pocet\_predpisani, rank() over(order by count(\*) desc) as poradie**

**from liek l join recept r on(l.id\_lieku = r.id\_lieku)**

**where to\_char(datum, 'YYYY') = '2020'**

**group by l.nazov, l.id\_lieku**

**) where poradie <= 0.10\*(select count(\*) from liek join recept using(id\_lieku) where to\_char(datum, 'YYYY') = '2020');**

Tento príkaz vráti 10% najviac predpisovaných liekov v roku 2022. V tomto príkaze sa pracuje s tabuľkami *liek* a *recept* a v základnom stave sa vykonáva za prítomnosti indexov nad všetkými PK a FK týchto tabuliek (v prípade receptu sú to indexy s jedným atribútom nad jednotlivými cudzími kľúčmi).

V tomto základnom stave je cost rovný 49. Plán vykonania bol takýto:

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Plán vykonania 4

Vidíme, že žiadny zo základných indexov sa pri vykonaní nepoužil. Napriek tomu, že v príkaze sa použité tabuľky joinujú dvakrát za dvoch rôznych okolností, reálne sa vykonal len jeden HASH JOIN, pretože na získanie hodnoty count(\*) nie je join potrebný, keďže z tabuľky *liek* nepotrebujeme získať žiadne dodatočné údaje, a taktiež by spojenie neovplyvnilo získaný počet riadkov, pretože každý recept sa musí viazať na konktrétny liek. Zmena číselného udania roku za reťazcovú formu ‘2022’ v podmienke where nemala účinok na výslednú hodnotu cost.

Následne sme vytvorili a testovali nasledujúce indexy:

Pre tabuľku *liek*:

create index LIEK0 on liek (nazov)

create index LIEK1 on liek (nazov, id\_lieku)

create index LIEK2 on lien (id\_lieku, nazov)

Pre tabuľku *recept*:

create index RECEPT1 on recept (to\_char(datum, 'YYYY')

create index RECEPT2 on recept (id\_lieku, to\_char(datum, 'YYYY'))

create index RECEPT3 on recept (to\_char(datum, 'YYYY'), id\_lieku)

create index RECEPT4 on recept (id\_receptu, id\_lieku, id\_pacienta, id\_lekara, to\_char(datum, 'YYYY'))

create index RECEPT4 on recept (to\_char(datum, 'YYYY'), id\_receptu, id\_lieku, id\_pacienta, id\_lekara)

Pri tvorbe indexov sme vyskúšali rôzne kombinácie atribútov a v rámci nich sme menili aj ich poradie a sledovali sme výsledný účinok na hodnotu cost.

V prípade indexov na tabuľku *liek*, mohli sme vzhľadom na malý počet atribútov tejto tabuľky (2) vyskúšať všetky možné indexy (index nad PK sa vytvoril implicitne). Tieto indexy ale nemajú na výkon príkazu veľký vplyv:

LIEK0 sa pri vykonávaní príkazu nepoužije

LIEK1 a LIEK2 sa pri vykonávaní použijú, pričom na poradí atribútov nezáleží, dôležitá je prítomnosť oboch atribútov. Index sa v tomto prípade použije namiesto samotnej tabuľky pri joinovaní oboch tabuliek, a využije sa metóda FAST FULL SCAN.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 7 Plán vykonania 6

Vplyv na výsledný cost je však minimálny, a jeho hodnota sa zníži o 2 body na 47.

Indexy nad tabuľkou *recept* mali na výsledné náklady príkazu oveľa väčší vplyv, pretože cielili na podmienku *where*. Hodnoty cost pri použití jednotlivých indexov sú uvedené v tabuľke:

Tabuľka Indexy recept

|  |  |
| --- | --- |
| INDEX | COST |
| RECEPT1 | 34 |
| RECEPT2 | 36 |
| RECEPT3 | 12 |
| RECEPT4 | 48 |
| RECEPT5 | 13 |

Vidíme, že poradie atribútov je v tomto prípade veľmi dôležité. Indexy RECEPT3 a RECEPT4 sa líšia len pozíciou atribútu *to\_char(datum, 'YYYY')*, ale keďže z hľadiska príkazu ide o dôležitý atribút, tak táto pozícia rozhoduje o využiteľnosti indexu.

Skombinovaním indexov nad oboma tabuľkami sme docielili náklady príkazu v hodnote 12, čo bola najnižšia hodnota, aká sa nám podarila dosiahnuť. Výsledný plán vykonania s použitím vytvorených indexov vyzeral nasledovne:

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 8 Plán vykonania 7

Najzásadnejší rozdiel oproti pôvodnému plánu vykonania bolo použitie indexu RECEPT3, pomocou ktorého je možné rýchlo pristupovať k tabuľke *recept* za pomoci metódy INDEX RANGE SCAN a tým zrýchliť join oboch tabuliek pri zadanej podmienke.

# Prístup k dátam na vzdialenom serveri

V našej semestrálnej práci pristupujeme k externým dátam, ktoré sa nachádzajú na školskom serveri asterix. Prepojenie sme vytvorili pomocou objektu database link. Tieto dáta predstavujú informácie o výplatách zamestnancov a nachádzajú sa v  tabuľke vyplata, ktorá slúži na uchovávanie výšok a dátumov výplat. Keďže tieto výplaty sa budú aktualizovať raz za mesiac rozhodli sme sa vytvoriť materializovaný pohľad vyplaty\_mw, ktorý v sebe tieto dáta uchováva a automaticky sa aktualizuje každý mesiac. Kód k vytvoreniu databázového linku a materializovaného pohľadu sa nachádza v priečinku DB\_Link. Štruktúru tabuľky vyplata je možné vidieť nižšie.

Obrázok, na ktorom je text, stôl

Automaticky generovaný popis

# Zoznam obrázkov

[Obrázok 1 Dátový model 3](#_Toc120979160)

[Obrázok 2 Zobrazenie štatistických výstupov 10](#_Toc120979161)

[Obrázok 3 Plán vykonania 1 12](#_Toc120979162)

[Obrázok 4 Plán vykonania 2 13](file:///C:\Users\PC1\Downloads\PDS_Dokumentacia.docx#_Toc120979163)

[Obrázok 5 Plán vykonania 3 13](#_Toc120979164)

[Obrázok 6 Plán vykonania 4 14](#_Toc120979165)

[Obrázok 7 Plán vykonania 6 15](file:///C:\Users\PC1\Downloads\PDS_Dokumentacia.docx#_Toc120979166)

[Obrázok 8 Plán vykonania 7 16](#_Toc120979167)

# Zoznam tabuliek

[Tabuľka 1 Indexy recept 16](#_Toc120979181)