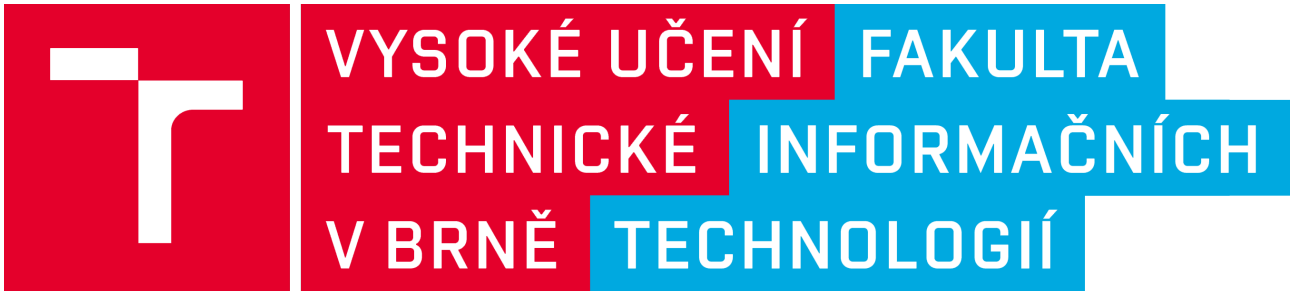


Databázové systémy

Projekt 1. časť - Dátový model (ERD), model prípadu využitia

29. dubna 2021

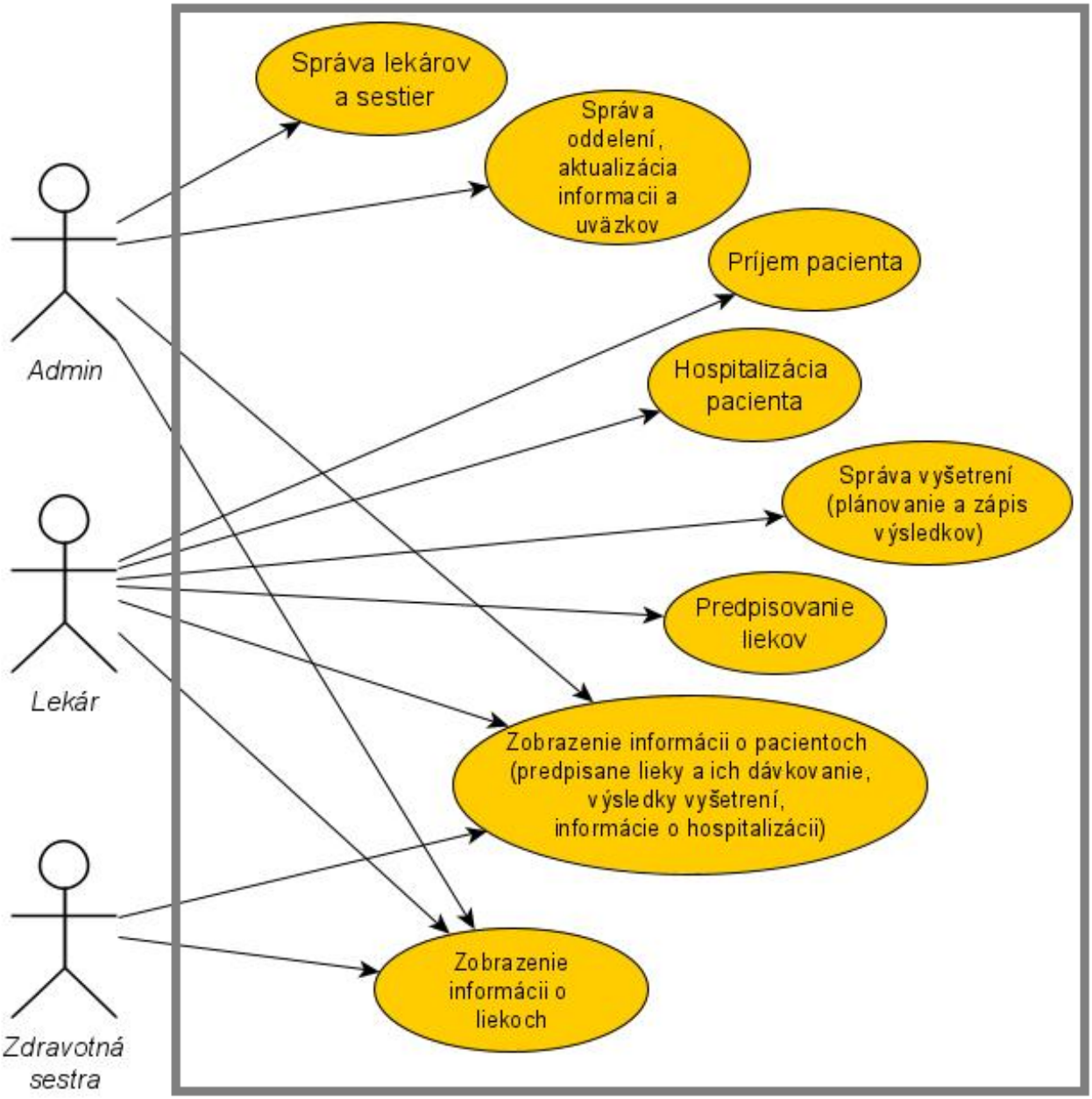
Matej Horník (xhorni20)
Filip Brna (xbrnaf00)

Obsah

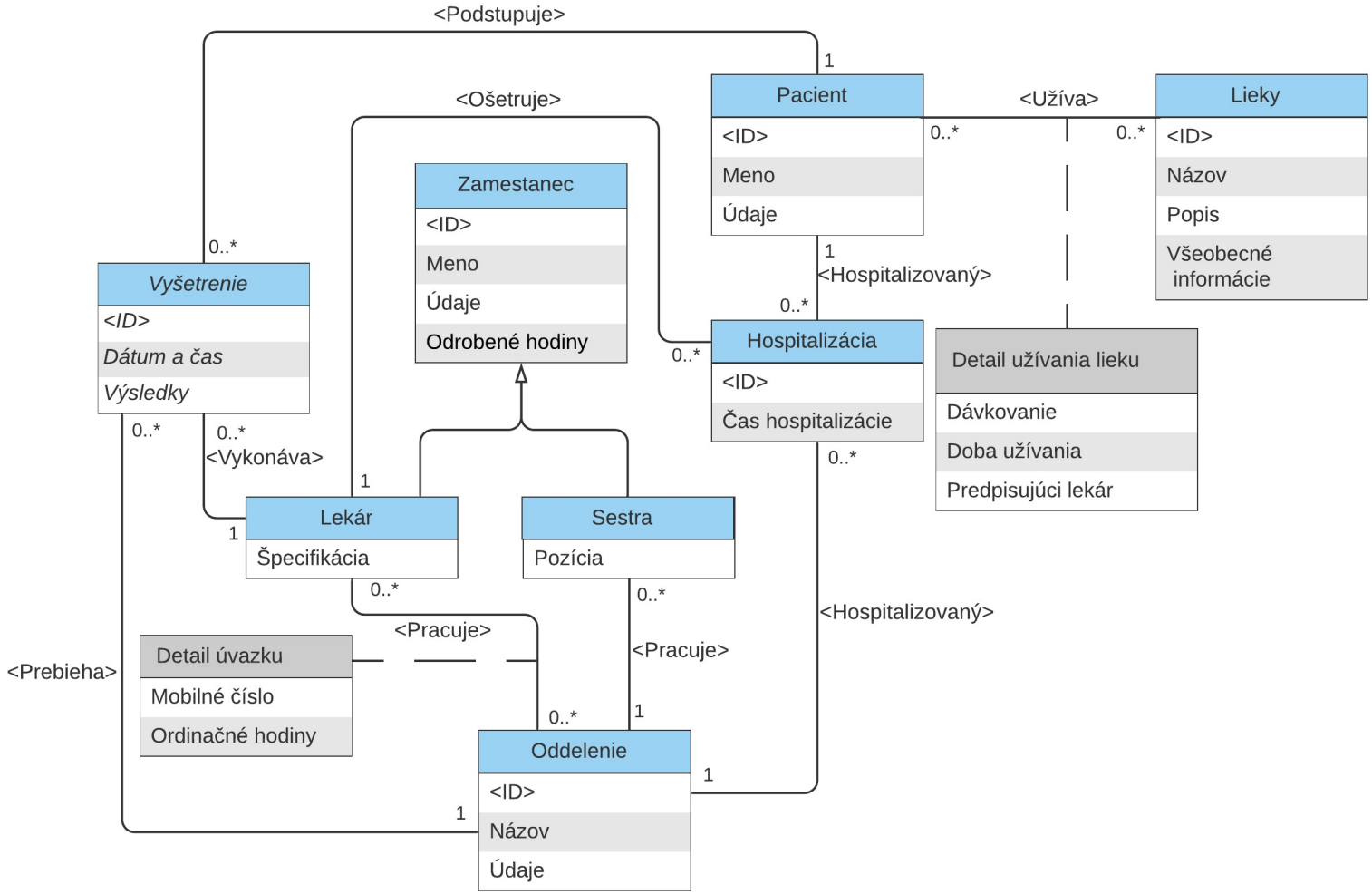
1 30. Zadanie (IUS) : Nemocnica

Navrhněte IS malé nemocnice, který by poskytoval základní údaje o lékařích, sestrách či pacientech, kteří jsou a byli hospitalizováni v nemocnici. IS uchovává informace o všech těchto hospitalizacích, přičemž pacient může být v jeden čas hospitalizován pouze na jednom oddělení nemocnice. Při každé hospitalizaci je mu určen jeho ošetřující lékař. Lékaři mohou pracovat na více odděleních zároveň. Na každém oddělení má lékař určitý úvazek, telefon atd., zatímco sestry pracují pouze na jednom oddělení. V rámci pobytu v nemocnici může pacient podstoupit různá vyšetření, která byla provedena na určitém oddělení ve stanoveném čase a provedl je určitý lékař, který také zapisuje výsledky vyšetření do IS. Dále mu mohou být podávány různé léky, každé podávání léku má určité detaily (kdy se podává, kolikrát apod.). V systému jsou uloženy i všeobecné informace o lécích (název, účinná látka, síla léku, kontraindikace atd.), aby si lékař mohl zkontrolovat správnost naordinovaného dávkování.

2 Use case diagram



3 ER diagram


<Podstupuje>
1
Pacient
<ID>
Meno
Údaje
<Užíva>
0..*
0..*
Lieky
<ID>
Názov
Popis
Všeobecné informácie
<Ošetruje>
Zamestanec
<ID>
Meno
Údaje
Odrobené hodiny
1
<Hospitalizovaný>
0..*
0..*
Hospitalizácia
<ID>
Čas hospitalizácie
Detail užívania lieku
Dávkovanie
Doba užívania
Predpisujúci lekár
0..*
Vyšetrenie
<ID>
Dátum a čas
Výsledky
0..*
0..*
<Vykonáva>
1
1
Lekár
Špecifikácia
Sestra
Pozícia
0..*
0..*
<Pracuje>
<Pracuje>
<Hospitalizovaný>
<Prebieha>
Detail úvazku
Mobilné číslo
Ordinačné hodiny
0..*
1
Oddelenie
<ID>
Názov
Údaje
1
1

Popis diagramu: V informačnom systéme (ďalej iba IS) sa budú uchovávať informácie o lekároch a sestrách, tieto dve entity sú špecializáciou entity zamestanca v ktorej sú uchované údaje o zamestnancoch. Lekári možu pracovať na viacerých oddeleniach, kde na každom oddelení majú svoje telefonne číslo a ordinačné hodiny, ďalej je v jeho náplni práce vykonávať vyšetrenia pacientov a ošetrovanie hospitalizovaných. Sestra pracuje iba na jednom oddelení. Ďalej v IS vystupuje aj pacient, ktorý može podstúpiť vyšetrenie. Vyšetrenie v IS predstavuje samostatnú entitu v ktorej sa uchovávajú informácie o danom vyšetrení, tieto vyšetrenia sa vykonávajú na určitom oddelení. Kedže pacient može byť počas svojho pobytu v nemocnici hospitalizovaný na viacerých oddeleniach, reprezentujeme hospitalizáciu ako entitu. Oddelenie okrem vyšetrení slúži aj na hospitalizáciu pacientov. V IS uchovávame všeobecné informácie o liekoch, ktoré užívajú pacienti spolu s dodatocnými informaciami (dávkovanie, doba užívania).

4 Vytvorenie objektov databázy

Tabuľky, z ktorých sa skladá nás databázový systém nemocnice:
– VYSETRENIE
s primárnym kľúčom ID_vyšetrenia a cudzím kľúčom rodné_číslo (pacient), ID_lekára (lekár), ICPE (oddelenie)

– LEKAR
s primárnym kľúčom ID_lekára

– SESTRA
s primárnym kľúčom ID_sestry s cudzím kľúčom ICPE (oddelenie)

– ODDELENIE
s primárnym kľúčom ICPE

– HOSPITALIZACIA
s primárnym kľúčom ID_hospitalizácie a cudzím kľúčom rodné_číslo (pacient), ID_lekára (lekár), ICPE (oddelenie)

– PACIENT
s primárnym kľúčom rodné_číslo

– LIEK
s primárnym kľúčom ID_lieku

– DETAIL UVAZKU
s primárnym kľúčom ID_lekára, ICPE a cudzím kľúčom ICPE (oddelenie), ID_lekára (lekár)

– DETAIL UZIVANIA LIEKU
s primárnym kľúčom rodné_číslo, ID_lieku a cudzím kľúčom rodné_číslo (pacient), ID_lieku (liek)

5 Implementácia

5.1 DROP

Aby nedochádzalo ku kolíziám je potrebné najprv zrušiť už vytvorené tabuľky pomocou príkazu DROP.

5.2 CREATE

Nasleduje vytvorenie všetkých potrebných tabuliek ich primárnych a cudzích kľúčov pomocou príkazu CREATE.

5.3 INSERT

Tabuľky je potrebné naplniť dátami pomocou príkazu INSERT, ktoré by predstavovali reálne dáta zadavane do tohto systému, aby bolo možné neskoršie simulovanie vytvorených príkazov, triggerov, procedúr,...

5.4 SELECT

V rámci 3 časti projektu sme mali implementovat niekoľko príkazov SELECT, tento prikaz slúži na vypísanie dát z tabuľky. Tieto selecty sme vytvarali aj pomocou nasledujúcich príkazov:

JOIN - spája záznamy z dvoch a viacerých tabuliek
GROUP BY - slúži pre agregáciu záznamov vyberaných pomocou príkazu SLELECT
HAVING - pomáha definovať omedzujucu podmienku
BETWEEN - Určuje či sa hodnota nachádza v otvorenom intervale medzi dvoma hraničnými hodnotami
NOT EXISTS - pokiaľ neexistuje to znamená, že je splnená podmienka NOT EXISTS tak potom je daný príkaz zobrazený

```
-- Ktory lekari vykonali viac ako jednu hospitalizaciu. ( ID_lekara, meno, priezvisko, pocet hospitalizacii

SELECT lek.ID_lekara, lek.meno, lek.priezvisko, COUNT(lek.ID_LEKARA) AS POCET_HOSPITALIZACII
FROM HOSPITALIZACIA hos
JOIN LEKAR lek ON lek.ID_lekara = hos.ID_lekara
GROUP BY lek.ID_lekara, lek.meno, lek.priezvisko
HAVING COUNT(lek.ID_lekara) > 1;
-- Ake su oddelenia na ktorych je presne jedna hospitalizacia v marci 2021. (ICPE, nazov oddelenia)

SELECT odd.ICPE, odd.nazov
FROM HOSPITALIZACIA hos
JOIN ODDELENIE odd ON odd.ICPE = hos.ICPE
WHERE hos.datum BETWEEN TO_TIMESTAMP('2021-03-01 00:00:01', 'YYYY-MM-DD HH24:MI:SS') AND TO_TIMESTAMP('2021-03-31 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
GROUP BY odd.ICPE, odd.nazov
HAVING COUNT(hos.ICPE) = 1;
-- ktory pacienti z mesta Brno podstupuju viacero vysetreni u jedneho lekara (rodne_cislo, meno, priezvisko, ulica, ID_lekara)

SELECT pac.rodne_cislo, pac.meno, pac.priezvisko, pac.ulica, lek.ID_lekara
FROM VYSETRENIE vys
JOIN LEKAR lek ON lek.ID_lekara = vys.ID_lekara
JOIN PACIENT pac ON pac.rodne_cislo = vys.rodne_cislo
WHERE pac.obec = 'Brno'
GROUP BY pac.rodne_cislo, pac.meno, pac.priezvisko, pac.ulica, lek.ID_lekara
HAVING COUNT(pac.rodne_cislo) > 1;
```

```sql
-- Ktory pacienti maju vysetrenia na vsetkych oddeleniach nemocnice. (rodne_cislo, meno, priezvisko)

SELECT pac.rodne_cislo, pac.meno, pac.priezvisko
FROM PACIENT pac
WHERE NOT EXISTS
    (SELECT * FROM ODDELENIE odd WHERE NOT EXISTS
        (SELECT * FROM VYSETRENIE vys WHERE vys.rodne_cislo=pac.rodne_cislo AND vys.ICPE=odd.ICPE));

-- Ktory pacienti boli na vyseterni len na oddeleni chirurgie (ICPE, nazov)

SELECT pac.rodne_cislo, pac.meno, pac.priezvisko
FROM PACIENT pac
WHERE pac.rodne_cislo NOT IN
    (SELECT pac.rodne_cislo
    FROM VYSETRENIE vys
    JOIN PACIENT pac ON pac.rodne_cislo = vys.rodne_cislo
    JOIN ODDELENIE odd ON odd.ICPE = vys.ICPE
    WHERE odd.nazov != 'chirurgicke')
AND pac.rodne_cislo IN
    (SELECT pac.rodne_cislo
    FROM VYSETRENIE vys
    JOIN PACIENT pac ON pac.rodne_cislo = vys.rodne_cislo
    JOIN ODDELENIE odd ON odd.ICPE = vys.ICPE
    WHERE odd.nazov = 'chirurgicke');
```

5.5 TRIGGER

Vytvorili sme dva netriviálne triggery,

1. Prvý trigger má podľa zadania za úlohu automatické generovanie hodnoty primárneho kľúča (PK). V našom prípade sa jedná o generovanie PK v tabuľke VYSETRENIE, kde je pri vložený alebo úprave prvku tabuľky VYSETRENIE automaticky vygenerovaná nová hodnota primárneho kľúča. Generovanie začína od 1, každá ďalšia vygenerovaná hodnota PK je zvačšená oproti predchádzajúcej o 1.

2. Druhý trigger je vytvorený na kontrolu správneho rodného čísla pacienta, táto kontrola nastáva v prípade, keď sú pridavané alebo aktualizované dáta v tabuľke PACIENT. V našej tabuľke sme si pre jednoduchosť zvolili validne rodné číslo také, že musí mať dĺžku 10 čísel a nemusí byť deliteľné 11. Naším triggerom, teda kontrolujeme či dané rodné číslo má správnu dĺžku (ak nie, je vyhodená chyba "Nesprávny formát rodného čísla") A či náhodou nie je zadaný nesprávny dátum napríklad rodné číslo 0004314567 nie je validné, pretože Duben (Apríl) má iba 30 dni (v tomto prípade je vyhodená chyba "Neplatný deň v mesiaci").

5.6 PROCEDURE

5.7 EXPLAIN PLAN a INDEX

Úlohou príkazu je zobrazenie postupnosti, ako sa budú dané operácie realizovať pre vybraný príkaz, taktiež vypíše informácie o výkonnostnej a časovej náročnosti danej operácie. Na prvom obrázku môžete vidieť plán vytvorený pomocou príkazu EXPLAIN PLAN na zadaný príkaz pre zistenie toho "Ktorý pacienti z mesta Brno podstúpili viacero vyšetrení u jedného lekára (rodné_číslo, meno, priezvisko, ulica, ID_lekára)"bez indexu.

```
PLAN_TABLE_OUTPUT
Plan hash value: 4135628773

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |     7 |   693 |     4  (25)| 00:00:01 |
|*  1 |  FILTER                       |            |       |       |            |          |
|   2 |   HASH GROUP BY               |            |     7 |   693 |     4  (25)| 00:00:01 |
|   3 |    NESTED LOOPS               |            |     7 |   693 |     3   (0)| 00:00:01 |
|   4 |     NESTED LOOPS              |            |     9 |   693 |     3   (0)| 00:00:01 |
|   5 |      TABLE ACCESS FULL        | VYSETRENIE |     9 |   225 |     3   (0)| 00:00:01 |
|*  6 |      INDEX UNIQUE SCAN        | PK_PACIENT |     1 |       |     0   (0)| 00:00:01 |
|*  7 |     TABLE ACCESS BY INDEX ROWID| PACIENT    |     1 |    74 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Na nasledujúcom obrázku môžete vidieť plán pre ten istý príkaz, avšak teraz už s použitím indexu obsahujucého ID_LEKARA a RODNE_CISLO pacienta. Táto zmena umožňuje efektívnejšie vykonanie príkazu, keďže namiesto prechodu celej tabuľky VYSETRENIE je použití nami vytvorený DOUBLE_INDEX.

```
PLAN_TABLE_OUTPUT
Plan hash value: 3757484097

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |     7 |   693 |     2  (50)| 00:00:01 |
|*  1 |  FILTER                       |              |       |       |            |          |
|   2 |   HASH GROUP BY               |              |     7 |   693 |     2  (50)| 00:00:01 |
|   3 |    NESTED LOOPS               |              |     7 |   693 |     1   (0)| 00:00:01 |
|   4 |     NESTED LOOPS              |              |     9 |   693 |     1   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | DOUBLE_INDEX |     9 |   225 |     1   (0)| 00:00:01 |
|*  6 |      INDEX UNIQUE SCAN        | PK_PACIENT   |     1 |       |     0   (0)| 00:00:01 |
|*  7 |     TABLE ACCESS BY INDEX ROWID| PACIENT      |     1 |    74 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Použitie indexovania je výhodou práve v prípade častého vyhľadávania v určitej tabuľke, avšak v prípade ak danú tabuľku často upravujeme nemusí byt indexovanie také efektívne, keďže je potrebné aktualizovať aj samotné indexy. V našom prípade často pristupujeme k tabuľke Lekár a Pacient takže sme pri vytváraní indexu použili práve stĺpce z týchto tabuliek, ktoré sú využité v príkaze SELECT (pacient RODNE_CISLO, lekár ID_LEKARA). Rozdiely medzi využívaním a nevyužívaním indexovania by boli väčšie ak by tabuľka obsahovala viac dát. Ďalšie zrýchlenie by sme mohli získať vytvorením druhého indexu, ktorý by bol vytvorený nad tabuľkou pacient a obsahoval by meno, priezvisko a ulicu pacienta.

5.8 Prístupové práva (GRANT)

Pridelenie prístupových práv druhému členovi tímu pomocou GRANT. Prístupové práva sú dôležitou súčasťou databázových systémov, každému užívateľovi môžeme zadať prístupové práva individuálne, na základe toho aké dáta budú pri výkone jeho práce potrebovať. Napríklad riaditeľ nemocnice, pre ktorú sme vytvárali tento databázový systém bude potrebovať väčšie práva ako zdravotná sestra pracujúca v nemocnici, pretože pre vykonávanie jej práce nieje potrebné, aby mala prístup ku všetkým dátam v databázy.

5.9 MATERIALIZED VIEW

Materializovany pohľad slúži na uloženie často využívaného pohľadu do pamäte. Hlavným účelom materializovaneho pohľadu je rýchly prístup k pohľadu pri opakovanom žiadaní.