Databázové systémy

2020/2021

Projektová dokumentácia

Nemocnica

Matúš Tvarožný (xtavro00)
Jakub Sokolík (xsokol14)

25.04.2021

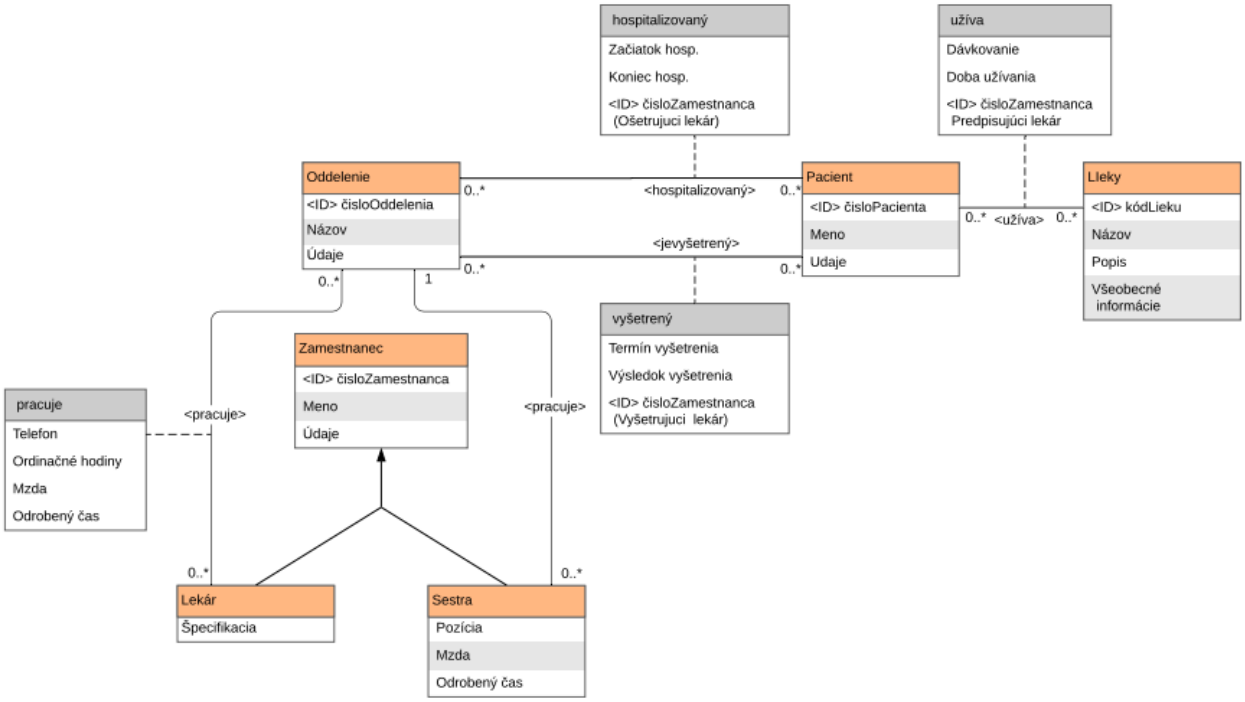
Obsah

1 Zadanie

Zadanie č. 30. z predmetu IUS – **Nemocnica**:

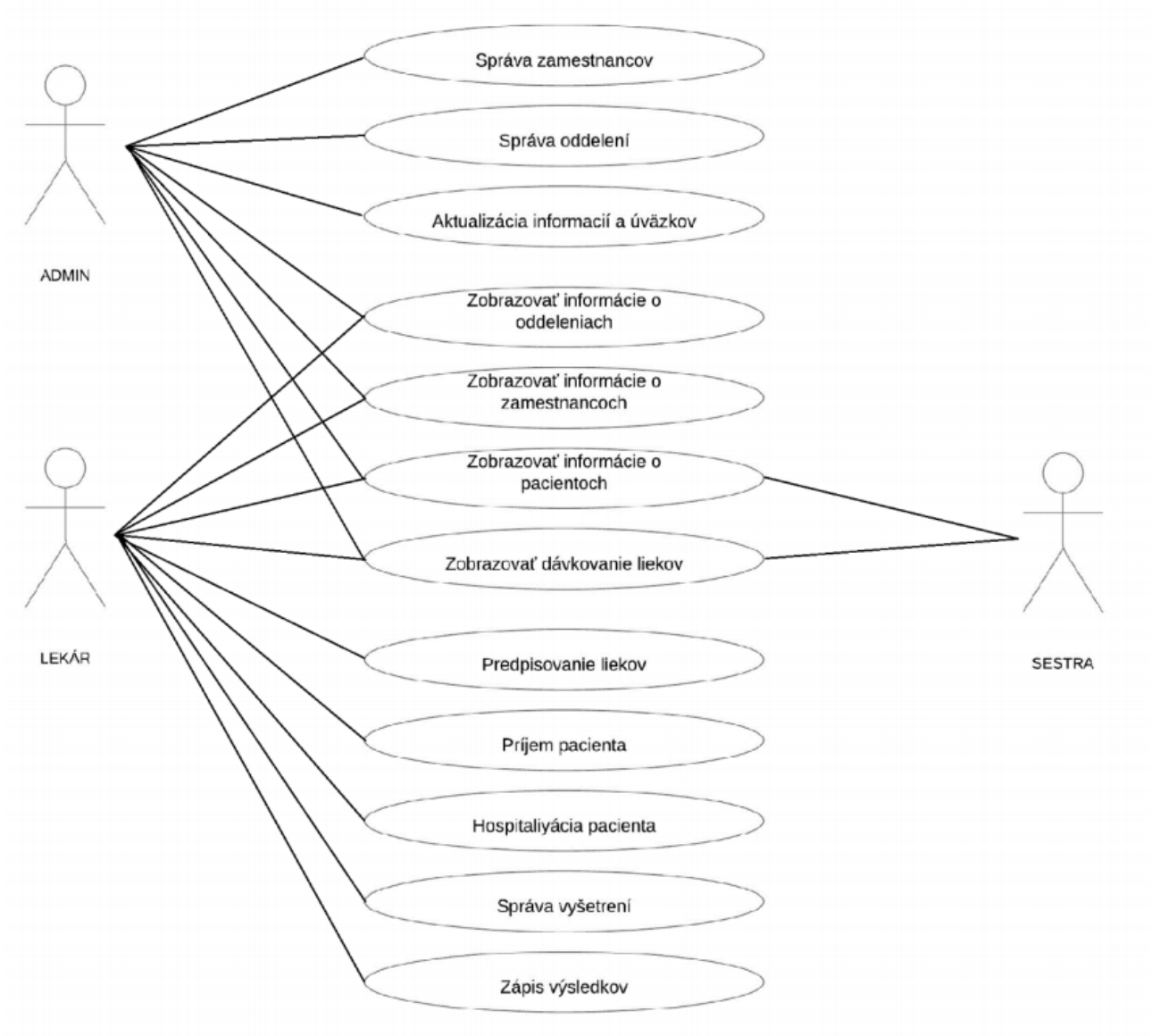
Navrhnete IS malé nemocnice, který by poskytoval základní údaje o lékařích, sestrách či pacientech, kteří jsou a byli hospitalizovaní v nemocnici. IS uchovává informace o všech těchto hospitalizacích, přičemž pacient muže být v jeden čas hospitalizován pouze na jednom oddělení nemocnice. Při každé hospitalizaci 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 podávaný různé léky, každé podávaní 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 dávkovaní.

2 ER diagram



obrázok 1. ER diagram

3 Use case diagram



obrázok 2. Use case diagram

4 Rozsah implementácie

1. **DROP** - Na začiatku nášho skriptu je sekcia DROP, ktorá slúži na zrušenie už existujúcich tabuliek, aby nedochádzalo k prípadným kolíziám.
2. **CREATE** - V tejto sekciu sú vytvorené tabuľky podľa zadania.
3. **TRIGGER** - V tejto sekciu sú vytvorené triggery podľa zadania projektu 4.
4. **INSERT** - V tejto sekciu sú tabuľky naplnené ukážkovými dátami.
5. **PROCEDURES** - V tejto sekciu sú vytvorené procedúry podľa zadanie projektu 4.
6. **EXPLAIN PLAN** - V tejto sekciu sú vytvorené indexy. Ich účinok na na rýchlosť vyhľadávania je ukázaný pomocou EXPLAIN PLAN podľa zadania projektu 4.
7. **MATERIALIZED WIEV** - V tejto je vytvorený pohľad podla zadania projektu 4.
8. **PROCEDURES** - V tejto sekcii sú definované prístupové práv k objektom databázy pre druhého člena týmu podľa zadania projektu 4.

5 Popis implementácie 4. časti projektu

5.1 Triggers

Našou úlohou bolo vytvoriť aspoň 2 triggery, z čoho jeden bude na generovanie primárneho kľúča.

1. trigger automaticky generuje primárny kľúč pre pacienta. Triger generuje kľúč ako počet záznamov tabuľke + 1. Tým je zaistené že kľúče budu nadväzovať na seba. Ak je tabuľka prázdna kľúč je automaticky 1.

```
CREATE OR REPLACE TRIGGER Id_generated
    BEFORE INSERT ON "patient"
    FOR EACH ROW
DECLARE
    tmp_id INT;
    tmp_cnt INT;
BEGIN
    IF :NEW."id" is NULL
    THEN
        SELECT count(*) INTO tmp_cnt
        From "patient" ;

        IF tmp_cnt = 0
        THEN
            :NEW."id" := 1;
        ELSE
            SELECT p."id" INTO tmp_id
            FROM "patient" p
            WHERE ROWNUM <= 1
            ORDER BY p."id" DESC;
            :NEW."id" := tmp_id + 1;
        END IF;
    END IF;
END;
```

id	birth_code	first_...	surename	ge...	birth_date
1	9908224832	Jakub	Sokolík	muž	22-AUG-99
2	9005084891	Ferko	Mrkvièka	muž	08-MAY-90
3	0002137325	Štefan	Koval	muž	13-FEB-00
4	7712122589	Jano	Sádecký	muž	12-DEC-77
5	9055084891	Janka	Novanská	žena	08-MAY-90
6	9858074369	Danka	Novanská	žena	07-AUG-90
7	9155084258	Klára	Staraková	žena	08-MAY-91

obrázok 3. Trigger na generovanie primárnych kľúčov a tabuľka pacientov

2. trigger kontroluje dátum narodenia pacienta. Ak je dátum zlý t.j. pacient sa ešte nenarodil, vypíše sa error.

```
CREATE OR REPLACE TRIGGER "check_birth_date"
    BEFORE INSERT ON "patient"
    FOR EACH ROW
DECLARE
    temp_date DATE;
BEGIN
    temp_date := TO_DATE(SYSDATE, 'YYYY-MM-DD');
    IF( :NEW."birth_date" > SYSDATE)
    THEN
        RAISE_APPLICATION_ERROR(-20000, 'Invalid date of birth.');
    END IF;
END;
```

obrázok 4.trigger na kontrolu veku

```
Error starting at line : 254 in command -
INSERT INTO "patient" ("birth_code", "first_name", "surename", "gender", "birth_date")
VALUES ('9908224832', 'Jakub', 'Sokolík', 'muž', TO_DATE('2022-08-22', 'YYYY-MM-DD'))
Error report -
ORA-20000: Invalid date of birth.
ORA-06512: at "XSOKOL14.check_birth_date", line 7
ORA-04088: error during execution of trigger 'XSOKOL14.check_birth_date'
```

obrázok 5. pokus o vloženie nenarodeneho pacienta

Pracovný úväzok vytváraný v tabuľke *work* obsahuje primárny kľúč zamestnanca, tento zamestnanec musí byť ale doktor. Z toho dôvodu sme vytvorili **3. trigger**, ktorý túto podmienku skontroluje

```
CREATE OR REPLACE TRIGGER Is_doctor
    BEFORE INSERT ON "work"
    FOR EACH ROW
DECLARE
    emp_type VARCHAR(10);
BEGIN
    SELECT e."type" INTO emp_type
    FROM "employee" e
    WHERE e."id" = :NEW."doctor_id";

    IF emp_type != 'doktor'
    THEN
        raise_application_error(-20101, 'ERROR: Vytvorit uvazok nie je mozne
                        pre sestru. Skontroluj ci si vlozil spravne doctor_id');
    END IF;
END;
```

obrázok 6. 3. trigger na kontrolu obmedzenia v tabuľke work

```
Error starting at line : 294 in command -
INSERT INTO "work" ("doctor_id", "depart_id", "phone", "work_time", "worked_hours", "salary")
VALUES ('8', '1', '+421090090090', 'po-str 6:30-14:00', '48', '16000')
Error report -
ORA-20101: ERROR: Vytvorit uvazok nie je mozne pre sestru. Skontroluj ci si vlozil spravne doctor_id
ORA-06512: at "XSOKOL14.IS_DOCTOR", line 10
ORA-04088: error during execution of trigger 'XSOKOL14.IS_DOCTOR'
```

obrázok 7. pokus o vloženie sestri do tabuľky work

Zo zadania vyplýva, že pacient v jednom čase nemôže ležať na 2 oddeleniach. Z toho dôvodu sme vytvorili **4. trigger**, ktorý toto obmedzenie kontroluje.

```
CREATE OR REPLACE TRIGGER "check_hospitalization"
    BEFORE INSERT ON "hospitalization"
    FOR EACH ROW
DECLARE
     temp INT;
BEGIN
    SELECT count(*) INTO temp
    FROM "hospitalization" h
    WHERE ((:NEW."start" >= h."start" AND :NEW."start" < h."end") OR
            (:NEW."end" > h."start" AND :NEW."end" <= h."end")
           OR (:NEW."start" <= h."start" AND h."start" < :NEW."end") OR
            (:NEW."end" > h."end" AND h."end" >= :NEW."start")) AND
            :NEW."patient_id" = h."patient_id";

    IF(temp > 0)
    THEN
        RAISE_APPLICATION_ERROR(-20000, 'Patient already laying on some
            department.');
    END IF;
END;
```

obrázok 8. 4. trigger na kontrolu hospitalizácie

```
Error starting at line : 299 in command -
INSERT INTO "hospitalization" ("patient_id", "doctor_id", "depart_id", "start", "end")
VALUES ('1', '2', '1', TO_DATE('2021-03-08', 'YYYY-MM-DD'), TO_DATE('2021-03-18', 'YYYY-MM-DD'))
Error report -
ORA-20000: Patient already laying on some department.
ORA-06512: at "XSOKOL14.check_hospitalization", line 11
ORA-04088: error during execution of trigger 'XSOKOL14.check_hospitalization'
```

obrázok 9. pokus o vloženie hospitalizacie s chybným časom

5.2 Procedúry

Našou úlohou bolo vytvoriť 2 procedúry, ktoré budú obsahovať kurzor, ošetrenie výnimiek a premennú s odkazom na typ.

1. procedúra vypočíta priemerný plat sestričiek. Po spustení procedúry dostaneme priemerný plat sestričiek *24 750*. Procedúra obsahuje ošetrenie výnimky ZERO_DIVIDE, pre prípad, že by tabuľka so sestričkami bola prázdna.

```
CREATE OR REPLACE PROCEDURE "average_income_nurse"
AS
    "average_income" NUMBER;
    "temp_income" NUMBER;
    "nurse_count" INT;
    CURSOR "cursor_average_salary" IS SELECT "salary" FROM "employee" WHERE "type" = 'sestra';
BEGIN
    SELECT COUNT(*) INTO "nurse_count"
    FROM "employee"
    WHERE "type" = 'sestra';

    "average_income" := 0;

    OPEN "cursor_average_salary";
    LOOP
        FETCH "cursor_average_salary" INTO "temp_income";
        EXIT WHEN "cursor_average_salary"%NOTFOUND;
        "average_income" := "average_income" + "temp_income";
    END LOOP;
    CLOSE "cursor_average_salary";

    DBMS_OUTPUT.put_line("average_income"/"nurse_count");

    EXCEPTION WHEN ZERO_DIVIDE THEN
    BEGIN
        IF "nurse_count" = 0 THEN
            DBMS_OUTPUT.put_line('Zero nurses in hospital.');
        END IF;
    END;
END;
```

obrázok 10. procedúra na výpočet priemerného platu sestričky

2. procedúra vypíše štatistiku oddelenia. Zameriava sa hlavne na to, koľko zo všetkých hospitalizácii a vyšetrení bolo vykonaných práve na danom oddelení. Procedúra obsahuje výnimku NO_DATA_FOUND pre prípad, že oddelenie na ktoré sa pýtame neexistuje.

```
CREATE OR REPLACE PROCEDURE Department_stat
    (depart_name IN VARCHAR)
AS
    all_hospit INT;
    all_exam INT;
    hospit_cnt INT;
    exam_cnt INT;
    depart_id "hospitalization"."depart_id"%TYPE;
    tmp_depart_id "hospitalization"."depart_id"%TYPE;
    CURSOR cursor_exam IS SELECT "depart_id" FROM "examination";
    CURSOR cursor_depart IS SELECT "depart_id" FROM "hospitalization";
```

obrázok 11. kúsok procedúry na získanie štatistík oddelenia

```
In Urgent department were 3 hospitalization from total count 5
In Urgent department were 3 examination from total count 9
```

obrázok 12. výpis procedúry po jej zavolaní na oddelenie urgent

5.3 Explain plan & index

```
SELECT p."first_name" AS meno, p."surename" AS priezvisko, COUNT(e."patient_id") AS pocet
FROM "patient" p
JOIN "examination" e ON e."patient_id" = p."id"
WHERE p."surename" LIKE 'S%'
GROUP BY p."first_name", p."surename"
HAVING COUNT(e."patient_id") > 1
ORDER BY pocet DESC, priezvisko, meno;
```

obrázok 13. skúmaný dotaz

Nad vyššie uvedeným dotazom sme spustili EXPLAIN PLAN aby sme zistili akým spôsobom bude dotaz prevedení. V na obrázku nižšie môžeme vidieť výsledok.

```
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |     7 |  1988 |     8  (25)| 00:00:01 |
|   1 |  SORT ORDER BY        |             |     7 |  1988 |     8  (25)| 00:00:01 |
|*  2 |   FILTER              |             |       |       |            |          |
|   3 |    HASH GROUP BY      |             |     7 |  1988 |     8  (25)| 00:00:01 |
|*  4 |     HASH JOIN         |             |     7 |  1988 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL| patient     |     3 |   813 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| examination |     9 |   117 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

obrázok 14. výpis plánu nad skúmaným dotazom

Počas riešenia dotazu sa pristupuje ku všetkým údajom z tabuľky pacient, čo zaberá pomerne veľa času. Vytvoríme preto index, ktorý bude obsahovať iba potrebné stĺpce pre náš dotaz.

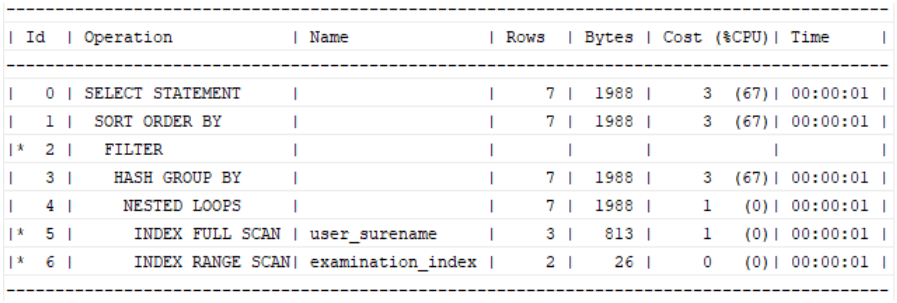
```
CREATE INDEX "user_surename" ON "patient" ("id", "first_name", "surename");
```

obrázok 15. vytvorený index na urýchlenie dotazu

```
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |               |     7 |  1988 |     6  (34)| 00:00:01 |
|   1 |  SORT ORDER BY        |               |     7 |  1988 |     6  (34)| 00:00:01 |
|*  2 |   FILTER              |               |       |       |            |          |
|   3 |    HASH GROUP BY      |               |     7 |  1988 |     6  (34)| 00:00:01 |
|*  4 |     HASH JOIN         |               |     7 |  1988 |     4   (0)| 00:00:01 |
|*  5 |      INDEX FULL SCAN  | user_surename |     3 |   813 |     1   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL| examination   |     9 |   117 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

obrázok 16. výpis plánu nad skúmaným dotazom s použitím indexu

Na obrázku 16. môžeme vidieť, že použitím indexu sme dosiahli zmenu. Nenačítavame všetky dáta z tabuľky pacient a využili sme vytvorený index. Vo výsledku sme ušetrili 2 jednotky ceny. Daná optimalizácia nie je zrovna veľká. To súvisí z tým že naša databáza je naplnená iba ukážkovými dátami. Ak by sme chce riešenie dotazu ešte viac optimalizovať mohli by sme vytvoriť ďalší index, ktorý by nám naindexoval tabuľku vyšetrení.



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		7	1988	3 (67)	00:00:01
1	SORT ORDER BY		7	1988	3 (67)	00:00:01
* 2	FILTER					
3	HASH GROUP BY		7	1988	3 (67)	00:00:01
4	NESTED LOOPS		7	1988	1 (0)	00:00:01
* 5	INDEX FULL SCAN	user_surename	3	813	1 (0)	00:00:01
* 6	INDEX RANGE SCAN	examination_index	2	26	0 (0)	00:00:01

obrázok 17. výpis plánu nad skúmaným dotazom s použitím 2 indexov

Ako môžeme vidieť, tak vytvorením indexu pre tabuľku vyšetrení sme dosiahli ďalšiu úsporu. Tento krát sa nám zmenil aj spôsob spojenia dát. Oproti predchádzajúcemu **HASH JOIN** sme teraz použili **NESTED LOOP**, čo nám prinieslo ďalšiu úsporu. Teraz vyriešenie celého dotazu trvá iba 3 jednotky ceny namiesto pôvodných 8.

5.4 prístupové práva

V našom prípade by sme mohli uvažovať o prístupových právach pre následujúcich aktérov:

1. **Správca** - človek zodpovedný za správu systému. Má prístup ku všetkému
2. **Doktor** - človek pracujúci s pacientmi. Má prístup K hospitalizáciám, vyšetreniam a dávkovaniu liekov.
3. **Sestra** - človek, ktorý sa stará o pacientov, ma prístup k informáciam o nich a k dávkovaniu liekov.

Prístupové práva reflektujú use case diagram. V implementácii je ukážka prístupových práv pre druhého člena týmu, čiže pre správcu.

5.5 Materialized wiev

Slúži na uloženie často využívaného pohľadu, za účelom rýchleho prístupu k tomuto pohľadu. V našom prípade sme sa rozhodli uložiť pohľad, pacientov a ich vyšetrení.

V našom prípade sa sleduj zmeny v tabuľkách pacient a vyšetrenie. Zmeny sa v danom pohľade uložia až po vykonaní ríkazu **COMMIT**

V implementácii sa nachádza ukážka použitia.