

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Databázové systémy

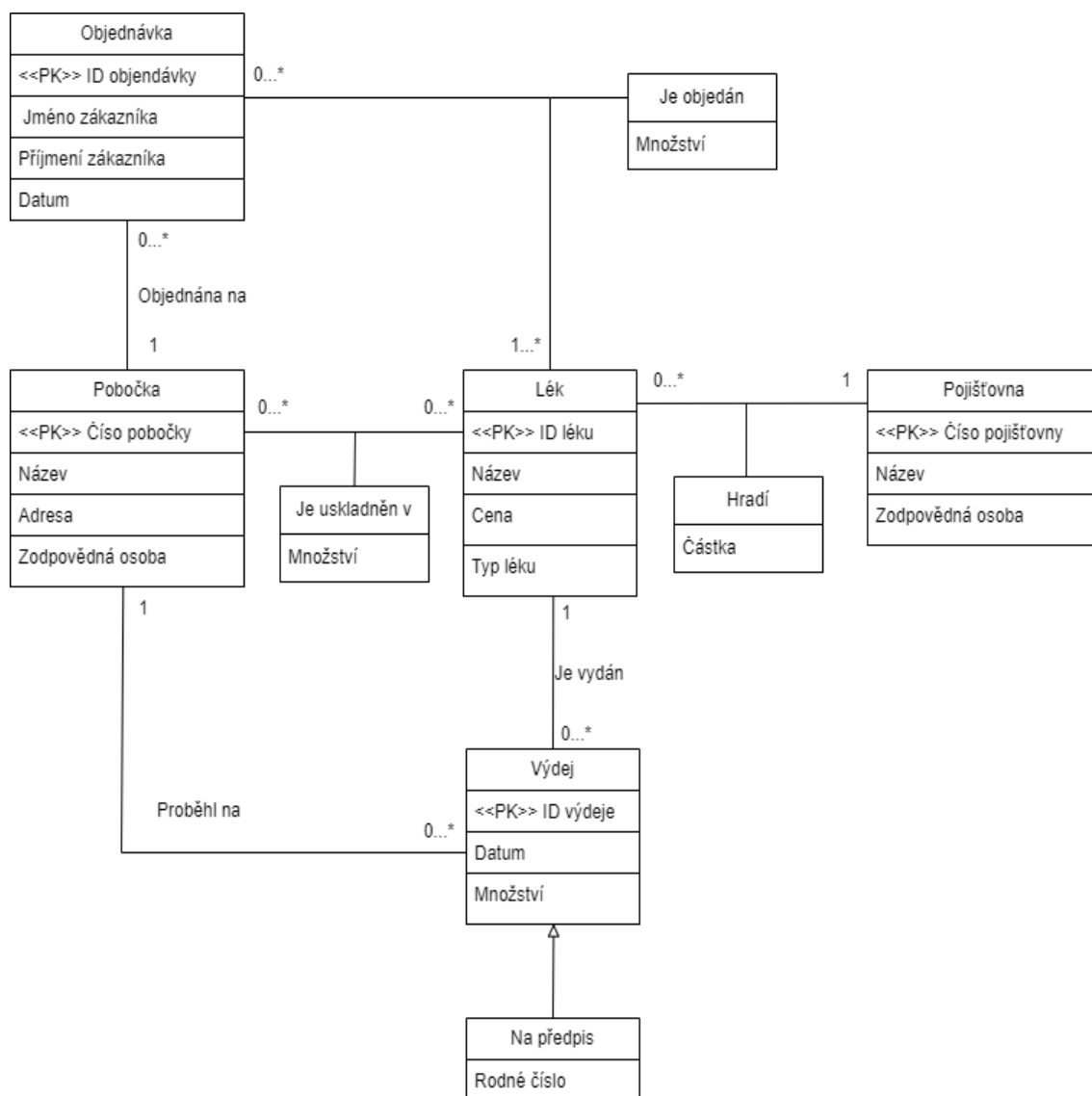
Zadání č. 33 - Lékárna

1 Úvod

Obsah

1	Úvod	1
2	Datový model	2
3	Popis řešení	3
3.1	Databázové triggery	3
3.2	Uložené procedury	4
3.3	Explain plan	5
3.4	Přístupová práva	5
3.5	Pohledy	6
3.5.1	Jednoduchý pohled	6
3.5.2	Materializovaný pohled	6

2 Datový model



Obrázek 1: Entity relationship diagram

3 Popis řešení

3.1 Databázové triggery

Trigger `KontrolaMnozstviNaSklade` je proveden při vložení dat do tabulky `Vydej`. Porovnává množství vyávaného léku a jeho množství na skladě na dané pobočce, odkud je výdej prováděn a pokud na skladě není dostatečné množství pro jeho vydání, dochází k chybě.

```
1 CREATE OR REPLACE TRIGGER KontrolaMnozstviNaSklade
2 BEFORE INSERT ON Vydej
3 FOR EACH ROW
4 DECLARE
5     mnozstvi_na_sklade NUMBER;
6 BEGIN
7     SELECT Mnozstvi INTO mnozstvi_na_sklade
8     FROM Uskladnen WHERE Cislo_pobocky = :NEW.Cislo_pobocky AND ID_leku = :NEW.ID_leku;
9     IF :NEW.Mnozstvi > mnozstvi_na_sklade THEN
10         RAISE_APPLICATION_ERROR(-20001, 'Daneho leku neni dostatecne mnozstvi!');
11     END IF;
12 END;
13 ;
14
```

Listing 1: SQL Trigger 1

Trigger `AktualizaceMnozstvi` je prováděn při vkládání dat do tabulky `Vydej`. Pokud dojde k vydání daného léku, je potřeba snížit množství tohoto léku na skladě. Proto tento trigger sníží aktuální množství na skladě na dané pobočce, odkud je lék vydáván, o jeho vydané množství.

```
1 CREATE OR REPLACE TRIGGER AktualizaceMnozstvi
2 AFTER INSERT ON Vydej
3 FOR EACH ROW
4 BEGIN
5     UPDATE Uskladnen
6     SET Mnozstvi = Mnozstvi - :NEW.Mnozstvi
7     WHERE Cislo_pobocky = :NEW.Cislo_pobocky AND ID_leku = :NEW.ID_leku;
8 END;
9 ;
10
```

Listing 2: SQL Trigger 2

Trigger `KontrolaCenyHrazeni` je prováděn při vkládání dat do tabulky `Hradi`. Pokud dojde ke vložení dat do této tabulky, reprezentující vztah, který udává, že daná pojišťovna může hradit část daného léku, je provedena kontrola, zdali hrazená částka nepřesahuje částku léku (nedává přeci smysl, aby pojišťovna hradila 200 Kč, když lék stojí 100 Kč).

```
1 CREATE OR REPLACE TRIGGER KontrolaCenyHrazeni
2 AFTER INSERT ON Hradi
3 FOR EACH ROW
4 DECLARE
5     cena_leku NUMBER;
6 BEGIN
7     SELECT Cena INTO cena_leku FROM Lek WHERE ID_leku = :NEW.ID_leku;
8     IF :NEW.Castka > cena_leku THEN
9         RAISE_APPLICATION_ERROR(-20002, 'Pojistovna nemuze hradit castku vyssi nez je
cena leku');
10    END IF;
11 EXCEPTION
12     WHEN NO_DATA_FOUND THEN
13         RAISE_APPLICATION_ERROR(-20003, 'Dany lek neexistuje v zaznamech!');
14 END;
15
```

Listing 3: SQL Trigger 3

3.2 Uložené procedury

Procedura `UtrzenaTrzba`, přijímající 1 parametr (výstupní) je navržena k výpočtu celkové tržby z prodeje léků. Tento výpočet se provádí na základě prodaného množství a ceny léku. Výstupní parametr `p_Trzba` uchovává tuto hodnotu, která je navracena volajícímu. Cursor zde slučuje tabulky `Vydej` a `Lek` podle `ID_leku` a slouží pro iteraci prodeje léků. `v_mnozstvi` a `v_cena` jsou lokální proměnné uchovávající množství a cenu z aktuálního záznamu kurzoru.

```
1 CREATE OR REPLACE PROCEDURE UtrzenaTrzba(p_Trzba OUT NUMBER)
2 AS
3     CURSOR cursor_prodeje IS
4         SELECT v.Mnozstvi, l.Cena
5         FROM Vydej v
6         JOIN Lek l ON v.ID_leku = l.ID_leku;
7     v_mnozstvi Vydej.Mnozstvi%TYPE;
8     v_cena Lek.Cena%TYPE;
9 BEGIN
10    p_Trzba := 0;
11    OPEN cursor_prodeje;
12    LOOP
13        FETCH cursor_prodeje INTO v_mnozstvi, v_cena;
14        EXIT WHEN cursor_prodeje%NOTFOUND;
15        p_Trzba := p_Trzba + (v_mnozstvi * v_cena);
16    END LOOP;
17    CLOSE cursor_prodeje;
18
19    EXCEPTION
20        WHEN NO_DATA_FOUND THEN
21            p_Trzba := 0;
22            DBMS.OUTPUT.PUT_LINE('Zadna data nebyla nalezena, celkova trzba je 0,-');
23        WHEN OTHERS THEN
24            DBMS.OUTPUT.PUT_LINE('Nastala neocekavana chyba: ' || SQLERRM);
25            IF cursor_prodeje%ISOPEN THEN
26                CLOSE cursor_prodeje;
27            END IF;
28 END;
29 ;
30
```

Listing 4: SQL Procedura 1

Procedura `AktualizaceSkladu` se vstupními parametry `p_cislo_pobocky`, `p_ID_leku` a `p_nove_mnozstvi` umožňuje změnu množství léku na skladě pro danou pobočku v případě, že dojde zboží od dodavatele. `v_stare_mnozstvi` je lokální proměnná obsahující aktuální stav množství léku (daného druhým parametrem) na skladě pro pobočku danou prvním parametrem. Pokud dodané množství (dáno třetím parametrem) je platné (není záporné ani 0), dojde k navýšení o toto množství. Chyba vzniká v případě nevidovaného léku, neplatného množství nebo neočekávané chyby, neovlivněné vstupními parametry.

```

1 CREATE OR REPLACE PROCEDURE AktualizaceSkladu(p_cislo_pobocky NUMBER, p_ID_leku
NUMBER, p_nove_mnozstvi NUMBER)
2 AS
3     v_stare_mnozstvi Uskladnen.Mnozstvi%TYPE;
4 BEGIN
5     SELECT Mnozstvi INTO v_stare_mnozstvi
6     FROM Uskladnen WHERE ID_leku = p_ID_leku AND Cislo_pobocky = p_cislo_pobocky
7     FOR UPDATE;
8     IF p_nove_mnozstvi > 0 THEN
9         UPDATE Uskladnen
10        SET Mnozstvi = Mnozstvi + p_nove_mnozstvi
11        WHERE ID_leku = p_ID_leku AND Cislo_pobocky = p_cislo_pobocky;
12    ELSE
13        RAISE_APPLICATION_ERROR(-20005, 'Nove mnozstvi nesmi byt zaporne nebo 0!');
14    END IF;
15 EXCEPTION
16    WHEN NO.DATA.FOUND THEN
17        RAISE_APPLICATION_ERROR(-20003, 'Dany lek neni v zaznamech o lecich!');
18    WHEN OTHERS THEN
19        RAISE_APPLICATION_ERROR(-20006, 'Nastala neocekavana chyba ' || SQLERRM);
20 END;
21 ;
22

```

Listing 5: SQL Procedura 2

3.3 Explain plan

```

1 EXPLAIN PLAN FOR
2 SELECT l.Nazev AS Nazev_Leku ,
3        SUM(v.Mnozstvi) AS Celkove_Vydano
4 FROM Lek l
5        JOIN
6        Vydej v ON l.ID_leku = v.ID_leku
7 GROUP BY l.Nazev;
8

```

Listing 6: SQL Explain plan

3.4 Přístupová práva

Přístupová práva jsou udělena následujícím způsobem:

```

1 GRANT ALL ON Pobocka TO xmacha86;
2 GRANT ALL ON Lek TO xmacha86;
3 GRANT ALL ON Objednavka TO xmacha86;
4 GRANT ALL ON Pojistovna TO xmacha86;
5 GRANT ALL ON Vydej TO xmacha86;
6 GRANT ALL ON VydejPredpis TO xmacha86;
7 GRANT ALL ON Uskladnen TO xmacha86;
8 GRANT ALL ON Hradi TO xmacha86;
9 GRANT ALL ON Objednan TO xmacha86;
10

```

Listing 7: SQL Přístupová práva

3.5 Pohledy

3.5.1 Jednoduchý pohled

Umožňuje získat přehled o jednotlivých výdejích provedených na daných pobočkách.

```
1 CREATE VIEW Objednavky AS
2 SELECT
3     p.Nazev AS Nazev_Pobocky ,
4     l.Nazev AS Nazev_Leku ,
5     o.Mnozstvi
6 FROM
7     Pobocka p
8 JOIN
9     Objednavka ob ON p.Cislo_pobocky = ob.Cislo_pobocky
10 JOIN
11     Objednan o ON ob.ID_objednavky = o.ID_objednavky
12 JOIN
13     Lek l ON o.ID_leku = l.ID_leku ;
14
```

Listing 8: SQL Pohled 1

3.5.2 Materializovaný pohled

Ukládá data reprezentující výpisy o prodejkách. Klauzule REFRESH určuje kdy a jak by se měl pohled obnovovat (v tomto případě je celý přepočítán při každém obnovení). ON DEMAND dále specifikuje, že k obnově dochází pouze při požadavku a ne automaticky.

```
1 CREATE MATERIALIZED VIEW ProdejniZprava
2 BUILD IMMEDIATE
3 REFRESH COMPLETE
4 ON DEMAND
5 AS
6 SELECT
7     l.Nazev AS Nazev_Leku ,
8     SUM(v.Mnozstvi) AS Celkove_Prodano ,
9     SUM(v.Mnozstvi * l.Cena) AS Trzba
10 FROM
11     Lek l
12 JOIN
13     Vydej v ON l.ID_leku = v.ID_leku
14 GROUP BY
15     l.Nazev ;
16
```

Listing 9: SQL Pohled 2