VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Databázové systémy

Zadání č. 33 - Lékárna

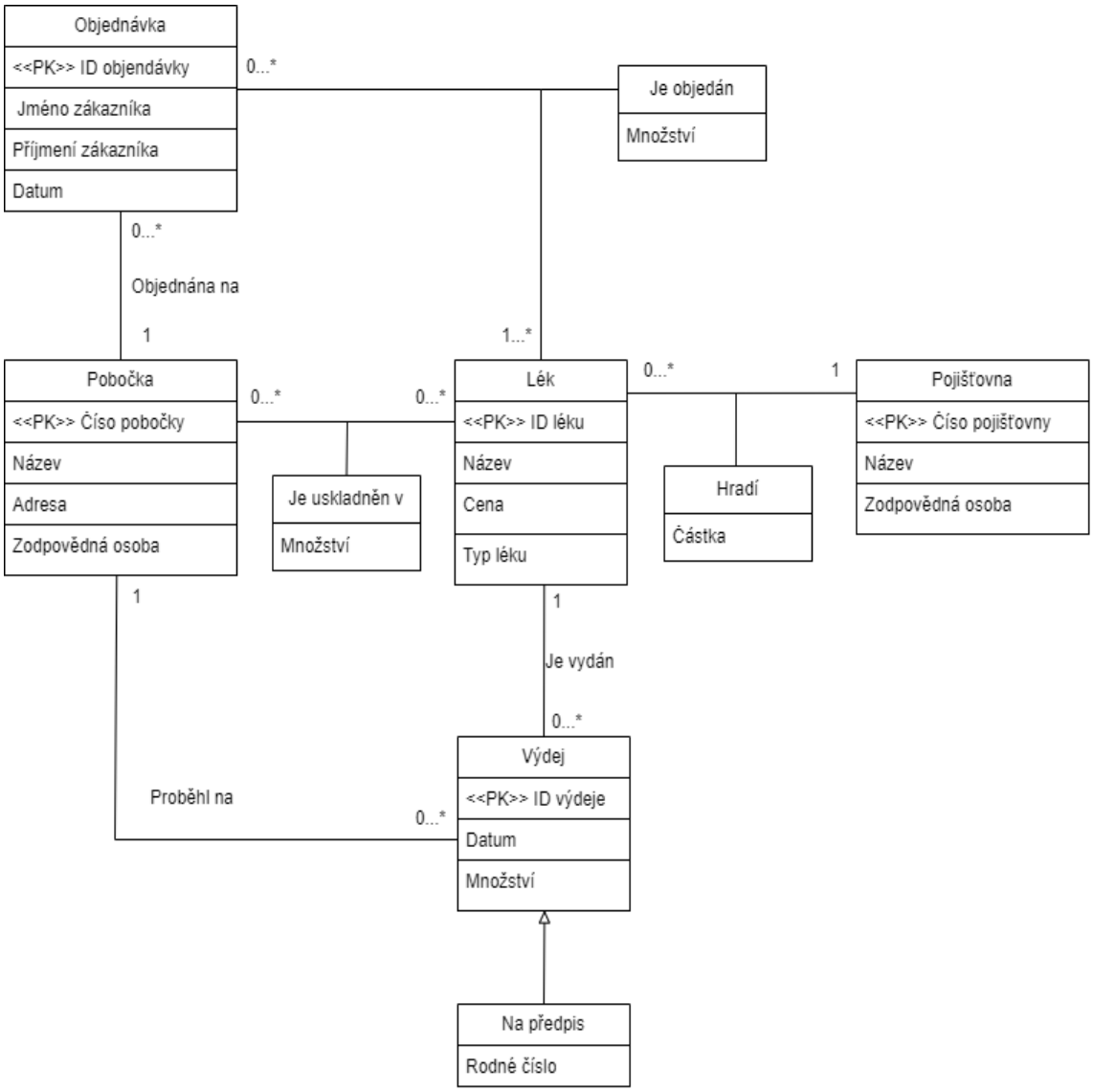
Adam Kán (xkanad00)
Roman Machala (xmacha86)

2024

1 Úvod

Obsah

2 Datový model



Obrázek 1: Entity relationship diagram

3 Popis řešení

3.1 Databázové triggery

Trigger `KontrolaMnozstviNaSklade` je proveden pri vlozeni 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.

```
        CREATE OR REPLACE TRIGGER KontrolaMnozstviNaSklade
        BEFORE INSERT ON Vydej
        FOR EACH ROW
        DECLARE
             mnozstvi_na_sklade NUMBER;
        BEGIN
             SELECT Mnozstvi INTO mnozstvi_na_sklade
             FROM Uskladnen WHERE Cislo_pobocky = :NEW.Cislo_pobocky AND ID_leku = :NEW.ID_leku;
             IF :NEW.Mnozstvi > mnozstvi_na_sklade THEN
                  RAISE_APPLICATION_ERROR(-20001, 'Daneho leku neni dostatecne mnozstvi!');
             END IF;
        END;
        ;

```

Listing 1: SQL Trigger 1

Trigger `AktualizaceMnozstvi` je prováděn při vkládání dat do tabulky `Vydej`. Pokud dojde k vydani daneho leku, 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.

```
        CREATE OR REPLACE TRIGGER AktualizaceMnozstvi
        AFTER INSERT ON Vydej
        FOR EACH ROW
        BEGIN
             UPDATE Uskladnen
             SET Mnozstvi = Mnozstvi - :NEW.Mnozstvi
             WHERE Cislo_pobocky = :NEW.Cislo_pobocky AND ID_leku = :NEW.ID_leku;
        END;
        ;

```

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).

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

```

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 navrácena 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 mnošství a cenu z aktuálního záznamu kurzoru.

```
CREATE OR REPLACE PROCEDURE UtrzenaTrzba(p_Trzba OUT NUMBER)
AS
    CURSOR cursor_prodeje IS
        SELECT v.Mnozstvi, l.Cena
        FROM Vydej v
        JOIN Lek l ON v.ID_leku = l.ID_leku;
    v_mnozstvi Vydej.Mnozstvi%TYPE;
    v_cena Lek.Cena%TYPE;
BEGIN
    p_Trzba := 0;
    OPEN cursor_prodeje;
    LOOP
        FETCH cursor_prodeje INTO v_mnozstvi, v_cena;
        EXIT WHEN cursor_prodeje%NOTFOUND;
        p_Trzba := p_Trzba + (v_mnozstvi * v_cena);
    END LOOP;
    CLOSE cursor_prodeje;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            p_Trzba := 0;
            DBMS_OUTPUT.PUT_LINE('Zadna data nebyla nalezena, celkova trzba je 0,-');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Nastala neocekavana chyba: ' || SQLERRM);
            IF cursor_prodeje%ISOPEN THEN
                CLOSE cursor_prodeje;
            END IF;
END;
/

```

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 lokalni promenna obsahující aktuální stav množství léku (daného druhým parametrem) na skladě pro pobočku danou prvním parametre. Pokud dodané množství (dáno třetím parametrem) je platné (není záporné ani 0), dojde k navýšení o toto množství. Chyba vzniká v případě neevidovaného léku, neplatného množství nebo neočekávané chyby, neovlivněné vstupními parametry.

```sql
CREATE OR REPLACE PROCEDURE AktualizaceSkladu(p_cislo_pobocky NUMBER, p_ID_leku NUMBER, p_nove_mnozstvi NUMBER)
AS
    v_stare_mnozstvi Uskladnen.Mnozstvi%TYPE;
BEGIN
    SELECT Mnozstvi INTO v_stare_mnozstvi
    FROM Uskladnen WHERE ID_leku = p_ID_leku AND Cislo_pobocky = p_cislo_pobocky
    FOR UPDATE;
    IF p_nove_mnozstvi > 0 THEN
        UPDATE Uskladnen
        SET Mnozstvi = Mnozstvi + p_nove_mnozstvi
        WHERE ID_leku = p_ID_leku AND Cislo_pobocky = p_cislo_pobocky;
    ELSE
        RAISE_APPLICATION_ERROR(-20005, 'Nove mnozstvi nesmi byt zaporne nebo 0!');
    END IF;
EXCEPTIO
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Dany lek neni v zaznamech o lecich!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20006, 'Nastala neocekavana chyba ' || SQLERRM);
END;
;

```

Listing 5: SQL Procedura 2

3.3 Explain plan

```sql
EXPLAIN PLAN FOR
SELECT o.Jmeno_zakaznika,
       o.Prijmeni_zakaznika,
       COUNT(*) AS Pocet_objednavek,
       SUM(l.Cena) AS Celkova_cena
FROM Objednavka o
         JOIN Lek l ON o.ID_leku = l.ID_leku
GROUP BY o.Jmeno_zakaznika, o.Prijmeni_zakaznika;

```

Listing 6: SQL Explain plan

3.4 Přístupová práva

Přístupová práva jsou udělena následujícím způsobem:

```sql
GRANT ALL ON Pobocka TO xmacha86;
GRANT ALL ON Lek TO xmacha86;
GRANT ALL ON Objednavka TO xmacha86;
GRANT ALL ON Pojistovna TO xmacha86;
GRANT ALL ON Vydej TO xmacha86;
GRANT ALL ON VydejPredpis TO xmacha86;
GRANT ALL ON Uskladnen TO xmacha86;
GRANT ALL ON Hradi TO xmacha86;
GRANT ALL ON Objednan TO xmacha86;

```

Listing 7: SQL Přístupová práva

3.5 Pohledy

3.5.1 Jednoduchý pohled

Umožňuje získat přehled o jednotlivých výdejích provedených na daných pobočkách.

```
CREATE VIEW Objednavky AS
SELECT
    p.Nazev AS Nazev_Pobocky,
    l.Nazev AS Nazev_Leku,
    o.Mnozstvi
FROM
    Pobocka p
JOIN
    Objednavka ob ON p.Cislo_pobocky = ob.Cislo_pobocky
JOIN
    Objednan o ON ob.ID_objednavky = o.ID_objednavky
JOIN
    Lek l ON o.ID_leku = l.ID_leku;

```

Listing 8: SQL Pohled 1

3.5.2 Materializovaný pohled

Ukládá data reprezentující výpisy o prodejích. 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.

```
CREATE MATERIALIZED VIEW ProdejniZprava
BUILD IMMEDIATE
REFRESH COMPLETE
ON DEMAND
AS
SELECT
    l.Nazev AS Nazev_Leku,
    SUM(v.Mnozstvi) AS Celkove_Prodano,
    SUM(v.Mnozstvi * l.Cena) AS Trzba
FROM
    Lek l
JOIN
    Vydej v ON l.ID_leku = v.ID_leku
GROUP BY
    l.Nazev;

```

Listing 9: SQL Pohled 2