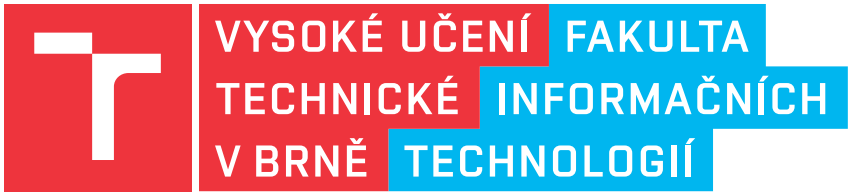


Databázové systémy – Projekt

Dátový model (ERD), model prípadov užitia

26. Banka

30. apríla 2022

Patrik Sehnoutek (xsehno01)
Dalibor Králik (xkrali20)

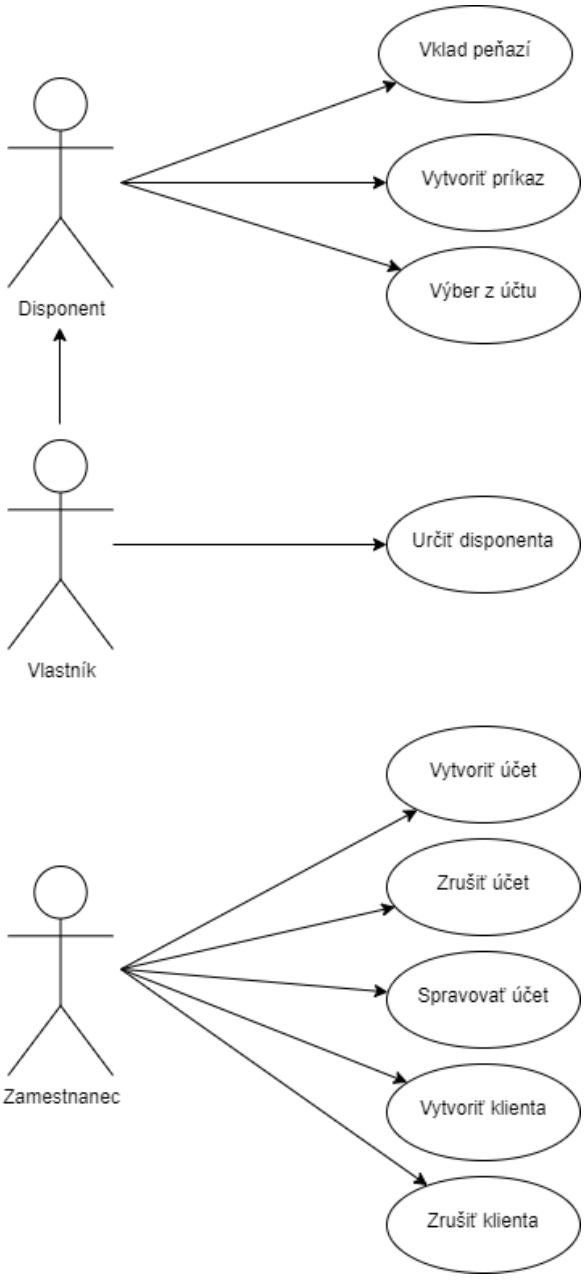
Obsah

1 Zadanie

Navrhnite modul informačného systému banky pre správu účtov. Banka poskytuje dva druhy účtom, sporiaci a bežný účet. O sporiacom účte bude v systéme informácia o úroku a o bežnom účte bude informácia o poplatku za vedenie účtu. O učte nesmú chýbať informácie ako číslo účtu, dátum založenia, zostatok a IBAN. Modul musí evidovať klientov, ich účty a operácie s nimi. Predpokladajte, že každý účet má jedného vlastníka, ale s účtom môže disponovať viacero osôb, ktoré určí vlastník. Disponent ma zadaný limit, s ktorým môže disponovať. Operácie zahrnujú vklad na účet, výber z účtu a prevod na iný účet (rovnakej alebo inej banky) sú realizované pomocou bankového príkazu, o ktorom banka udržuje dáta, napríklad poradové číslo, dátum, čiastka a typ príkazu. Systém musí ukladať informácie o všetkých operáciách s účtom (kto zadal, kedy, aká operácia a čiastka, kto vykonal). So systémom vždy priamo komunikuje iba zamestnanec banky, ktorý môže vytvárať účty, rušiť účty, vytvárať nových klientov v systéme a rušiť klientov v systéme. O zamestnancovi bude v systéme evidované jeho základné informácie ako meno, priezvisko, bydlisko, kontaktné údaje a dátum narodenia. Systém musí tiež okrem iného poskytovať výpis z účtu, ktorý sa posiela vlastníkovi, tj. výpis všetkých operácií s účtom za dané obdobie.

2 Diagram prípadov užitia (Use Case Diagram)

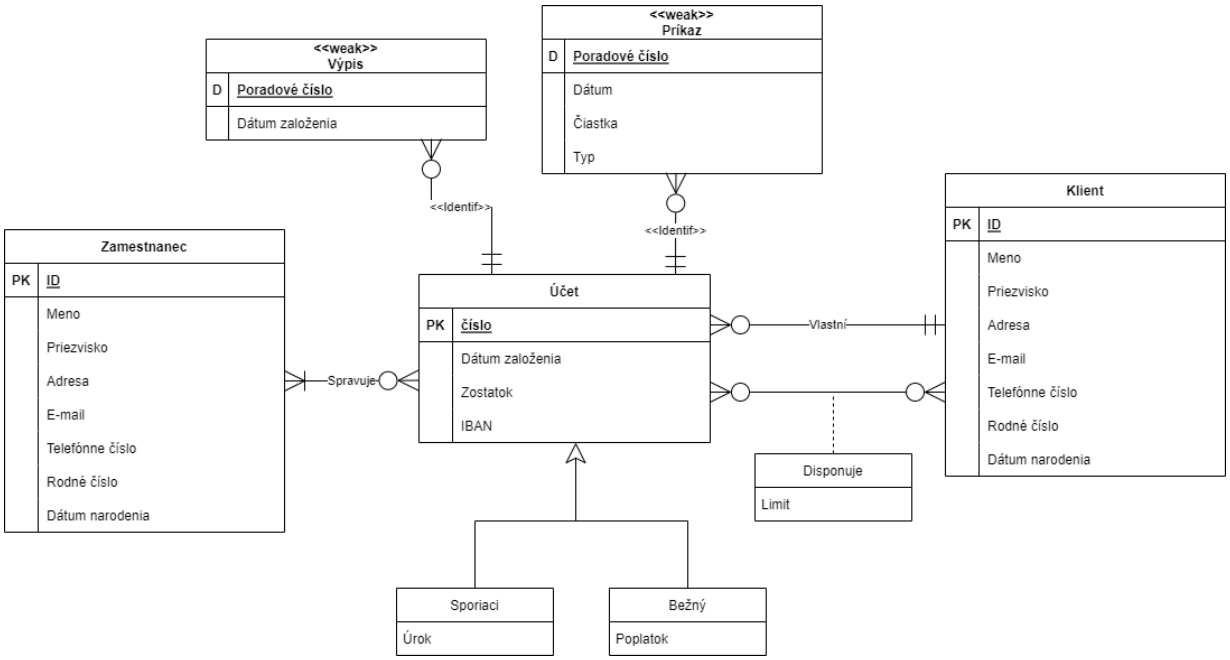
Model prípadov užitia pozostáva z troch užívateľov: disponenta, vlastníka a zamestnanca. Vlastník a disponent predstavujú klienta banky. Zamestnanec ako privilegovaný užívateľ má na starosti administráciu účtov klientov, čo v sebe zahŕňa vytváranie, rušenie a správu účtov. Taktiež je jeho úlohou zakladanie a rušenie účtov samotných klientov. Ďalším užívateľom je disponent, ktorý má právo vytvárať bankové príkazy a manipulovať so zostatkom na účte: vklad a výber financií. Vlastník má rovnaké práva ako disponent, navyše však môže určiť disponentov svojho účtu.



Obr. 1: Diagram prípadov užitia (Use Case Diagram)

3 Dátový model (ERD)

Navrhnutý ER diagram obsahuje niekoľko silných a slabých entít. Entita **KLIENT** vlastní **ÚČET** a disponuje na ňom určitým limitom. Vlastníkom účtu môže byť iba jeden klient, ale disponovať ním môže niekoľko ďalších klientov, ktorých určí vlastník. Disponovanie s účtom v sebe zahŕňa tvorbu platobných **PRÍKAZOV** a možnosť **VÝPISU**, aby mal klient prehľad o zostatku a pohybe peňazí na účte. Tieto dve spomínané entity nemôžu existovať bez toho, aby mal užívateľ založený účet. Účet sa rozdeľuje na dva typy, **SPORIACI** a **BEŽNÝ**. Za bežný účet je vedený poplatok a sporiaci účet zahŕňa určitý úrok. Účet je spravovaný povereným **ZAMESTNANCOM** banky, ktorý môže mať na starosti niekoľko účtov.



Obr. 2: Dátový model (ERD)

4 Vytvorenie základných objektov schémy databázy

4.1 Vytvorenie tabuliek

PRIKAZ
VYPIS
SPRAVUJE
ZAMESTNANEC
DISPONUJE
UCET
KLIENT

4.2 Naplnenie tabuliek vzorovými dátami

Tabulky sme naplnili vzorovými dátami pomocou príkazov `INSERT INTO` a `VALUES`. Prvý príkaz slúži na vybratie tabuľky a špecifikáciu stĺpcov tabuľky. Do druhého príkazu sa pridávajú už samotné dáta.

```
    INSERT INTO KLIENT( meno, priezvisko, ulica, mesto, email, telefon, rodne_cislo,
datum_narodenia)
    VALUES('Jan', 'Novak', 'Staromestska', 'Praha', 'jan@centrum.cz', '445322564', '
011210/3356', TO_DATE('2001-12-10', 'YYYY-MM-DD'));
```

5 Výber dát pomocou dotazov `SELECT`

V tejto sekcii sme spracovali prehľad jednotlivých dotazov na databázu. Pod každou ukážkou kódu je napísané presne, čo vykonáva.

5.1 Spojenie 2 tabuliek

```
SELECT meno, priezvisko
FROM KLIENT NATURAL JOIN UCET
WHERE urok IS NOT NULL;
```

Listing 1: Výpis klientov, ktorí majú sporiaci účet

```
SELECT DISTINCT meno, priezvisko
FROM ZAMESTNANEC NATURAL JOIN UCET
WHERE poplatok IS NOT NULL;
```

Listing 2: Výpis zamestnancov, ktorí spravujú bežný účet

5.2 Spojenie 3 tabuliek

```
        SELECT meno, priezvisko, c_uctu
        FROM KLIENT NATURAL JOIN UCET JOIN VYPIS V using(c_uctu)
        WHERE V.datum_zalozenia BETWEEN TO_DATE('2022-02-01', 'YYYY-MM-DD') AND
TO_DATE('2022-03-31', 'YYYY-MM-DD');
```

Listing 3: Ktorí klienti a na akom účte vytvorili výpis z účtu za obdobie 1.2.2022 - 31.3.2022

5.3 GROUP BY a agregačná funkcia SUM

```
SELECT meno, priezvisko, SUM(limit) celkovy_limit
FROM KLIENT join DISPONUJE USING(ID_klient)
GROUP BY meno, priezvisko
ORDER BY celkovy_limit DESC;
```

Listing 4: Jednotliví klienti a suma, ktorou disponujú na všetkých účtoch, zoradení zostupne

5.4 GROUP BY a agregačná funkcia COUNT

```
SELECT meno, priezvisko, COUNT(*)
FROM ZAMESTNANEC NATURAL JOIN SPRAVUJE
GROUP BY meno, priezvisko;
```

Listing 5: Počet účtov, ktoré spravujú jednotliví zamestnanci

5.5 Predikát IN s vnoreným príkazom SELECT

```
SELECT meno, priezvisko
FROM KLIENT
WHERE ID_klient IN
    (SELECT ID_klient FROM UCET
        WHERE c_uctu IN
        (SELECT c_uctu FROM PRIKAZ
            WHERE ciastka >= 500));
```

Listing 6: Ktorí klienti vytvorili príkaz s hodnotou minimánle 500€

5.6 EXISTS

```
SELECT K.meno, K.priezvisko
FROM KLIENT K, UCET U
WHERE K.ID_klient=U.ID_klient AND urok IS NOT NULL
      AND EXISTS(SELECT *
          FROM UCET U
          WHERE K.ID_klient=U.ID_klient AND
                U.poplatok IS NOT NULL);
```

Listing 7: Ktorí klienti sú vlastníkmi sporiaceho aj bežného účtu

6 Nastavenie prístupových práv

Prístupové práva boli nastavené vedúcim tímu (xsehno01) pre druhého člena (xkrali20). Na všetky vytvorené tabuľky sú nastavené práva pomocou `GRANT ALL ON` a na vytvorené procedúry pomocou `GRANT EXECUTE ON`.

7 Vytvorenie pokročilých objektov schémy databázy

7.1 Triggery

Implementovali sme nasledovné dva triggery `Zmena_uroku` a `Zmena_formatu_rodneho_cisla`.

7.1.1 Zmena_uroku

Trigger sa zavolá pred vložením alebo aktualizovaním zostatku v tabulke UCET. Úrok na sporiacom účte sa navýši, ak má klient na tomto účte zostatok aspoň 100 000.

7.1.2 Zmena_formatu_rodneho_cisla

Trigger, ktorý sa zavolá pred vkladaním alebo aktulizovaním rodného čísla. V prípade zlého formátu, tj. chýbajúceho lomítka toto lomítko vloží pre zachovanie správneho a očakávaného formátu rodného čísla

7.2 Materializovaný pohľad

Pohľady, u ktorých je výsledok dotazu definujúceho pohľad skutočne fyzicky uložený v databázi (na disku) za účelom rýchleho prístupu pri opakovanom žiadaní o tento pohľad. Je zaistená aktualizácia obsahu.

V našom kóde sme implementovali materializovaný pohľad `Pocet_uctov_zamestnancov` pre zobrazenie jednotlivých zamestnancov a počtu účtov, ktoré spracovávajú. Zvolili sme tento pohľad hlavne kvôli

tomu, aby v banke mohli ľahko vypísať, ako sú vyťažení jednotliví zamestnanci a komu prideliť ďalších klientov (účty). Poprípade ako prerozdeliť súčasných klientov (účty). Na aktualizáciu obsahu pohľadu po zmenách na tabuľkách sa musí spustiť príkaz `COMMIT`.

7.3 Procedúry

Vytvorili sme dve procedúry `Trasakcie_na_bankovom_ucte` a `Klienti_narodeniny`. Procedúry sme následne takto spúštali a ich správnosť sme skontrolovali manuálne pomocou výpisu tabuľky a overenia potrebných položiek.

```
begin
Transakcie_na_bankovom_ucte('1234567987/0300');
end;

begin
Klienti_Narodeniny();
end;
```

Listing 8: Spúšťanie procedúr

7.3.1 Trasakcie_na_bankovom_ucte

V procedúre je využitý kurzor, datový typ odkazujúci na riadok a ošetrenie výnimiek.

Procedúra má jeden parameter, ktorým je číslo bankového účtu a pracuje s tabuľkou `PRIKAZ`. Na konci sa pre daný účet vypíše celkový súčet príjmov a celkový súčet výdavkov na účte od založenia účtu. Za príjmy sa považuje typ príkazu `VKLAD`.

7.3.2 Klienti_narodeniny

V procedúre je využitý kurzor a datový typ odkazujúci na stĺpec.

Procedúra nemá žiadne parametre a pracuje iba s tabuľkou `KLIENT`. Prejde riadok po riadku a vypíše všetkých klientov, ktorí majú dnes narodeniny. Na dnešný dátum sa využíva zabudovaná funkcia `sysdate`. Následne sa porovnávajú mesiac a deň dátumu narodenia klienta a dnešného dátumu pomocou `substr`.

7.4 EXPLAIN PLAN a využitie indexu

Výpis klienta, čísla účtu a počtu príkazov, ktoré boli nad účtami klienta prevedené a celková suma čiastok príkazov na danom účte.

```
SELECT ID_klient, meno, priezvisko, c_uctu, count(*), sum(ciastka)
FROM KLIENT natural join UCET join PRIKAZ using(c_uctu)
GROUP BY ID_klient, meno, priezvisko, c_uctu;
```

Listing 9: Použitý príkaz

Vytvorili sme nasledovné indexy. Prvý index sme vytvorili kvôli príkazu `GROUP BY`, keďže potrebujeme prístup v tabuľke `KLIENT` iba ku stĺpcom v ňom špecifikovaným. Druhý index sa zameriava na tabuľku `UCET`, z ktorej potrebujeme iba dva stĺpce. Vytvoreným indexom sme chceli dosiahnút zrýchlenie prístupu k daným tabuľkám.

```
CREATE INDEX KLIENT_INDEX on KLIENT(ID_klient, meno, priezvisko);
CREATE INDEX UCET_INDEX on UCET(ID_klient, c_uctu);
```

Listing 10: Použitý príkaz

Na obrázku nižšie môžeme po zavolaní `EXPLAIN PLAN` vidieť, jednotlivé vykonané operácie. Posledné operácie sú prístup k potrebným dátam na vykonanie daného dotazu. Môžeme vidieť, že sa pristupuje k celým tabuľkám `TABLE ACCESS FULL`.

```
3  ------------------------------------------------------------------------------
4  | Id  | Operation           | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
5  ------------------------------------------------------------------------------
6  |   0 | SELECT STATEMENT    |        |    13 |  1313 |    10  (10)| 00:00:01 |
7  |   1 |  HASH GROUP BY      |        |    13 |  1313 |    10  (10)| 00:00:01 |
8  |*  2 |   HASH JOIN         |        |    13 |  1313 |     9   (0)| 00:00:01 |
9  |*  3 |    HASH JOIN        |        |     9 |   711 |     6   (0)| 00:00:01 |
10 |   4 |     TABLE ACCESS FULL| KLIENT |     8 |   456 |     3   (0)| 00:00:01 |
11 |   5 |     TABLE ACCESS FULL| UCET   |     9 |   198 |     3   (0)| 00:00:01 |
12 |   6 |    TABLE ACCESS FULL | PRIKAZ |    13 |   286 |     3   (0)| 00:00:01 |
13 ------------------------------------------------------------------------------
```

Obr. 3: Výsledky spustenia EXPLAIN PLAN pred vytvorením indexu.

Po aplikovaní indexu môžeme vidieť, že už sa nepristupuje ku celým tabuľkám, ale iba k jednotlivým indexom (stĺpcom, ktoré sú definované indexom), čo urýchľuje vykonanie dotazu.

```
3  ------------------------------------------------------------------------------------
4  | Id  | Operation           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
5  ------------------------------------------------------------------------------------
6  |   0 | SELECT STATEMENT    |              |    13 |  1313 |     6  (17)| 00:00:01 |
7  |   1 |  HASH GROUP BY      |              |    13 |  1313 |     6  (17)| 00:00:01 |
8  |*  2 |   HASH JOIN         |              |    13 |  1313 |     5   (0)| 00:00:01 |
9  |*  3 |    HASH JOIN        |              |     9 |   711 |     2   (0)| 00:00:01 |
10 |   4 |     INDEX FULL SCAN | KLIENT_INDEX |     8 |   456 |     1   (0)| 00:00:01 |
11 |   5 |     INDEX FULL SCAN | UCET_INDEX   |     9 |   198 |     1   (0)| 00:00:01 |
12 |   6 |    TABLE ACCESS FULL| PRIKAZ       |    13 |   286 |     3   (0)| 00:00:01 |
13 ------------------------------------------------------------------------------------
```

Obr. 4: Výsledky spustenia EXPLAIN PLAN po vytvorení indexu.