Zadaní: Informační systém lékárny
Autoři: Jakub Zapletal (xzaple36), Martin Studený (xstude23)

Tabulky:

- Pobocka (<<PK>> ID_pobocky, Mnozstvi_zbozi, Prodane_zbozi, Adresa)
- Lek (<<PK>>Ciselny_kod, Nazev, Cena)
- Na_predpis (<<PK>><<fk>>ID_leku, ID_predpisu, RC_zakaznika, ID_pojistovny)
    - Tvar RC_zakaznika je kontrolován regulárním výrazem[1]
    - Jedná se o specializaci léku
- Pojistovna (<<PK>>ICO_pojistovny, Nazev)
    - Tvar ICO_pojistovny je kontrolován pomocí aritmetického výpočtu[2]
- Dodavatel (<<PK>>ICO_dodavatele, Nazev, <<FK>>Dodavany_lek)
    - Tvar ICO_ dodavatele je kontrolován pomocí aritmetického výpočtu[2]
- Dodavka (<<PK>>ID_dodavky, <<FK>>ID_leku, <<FK>>ID_pobocky, <<FK>> ICO_dodavatele)
- Prispela (<<PK>>(Prispevek, <<FK>>id_lek, <<FK>>id_pojistovna))
- Ma_v_sortimentu (<<PK>>(<<FK>>ID_leku, <<FK>>ID_pobocky))
- Rezervace (<<PK>>ID_rezervace, Jmeno_zakaznika, Datum, <<FK>>ID_pobocky, <<FK>>ID_leku)

Triggery:

- auto_id_rezervace
    - Nastaví unikátní hodnotu primárního klíče pokud nebyl při vkládání do tabulky Rezervace specifikován.
    - Klíč je generován tak, že je nalezeno maximum z celočíselných klíčů v tabulce a nový klíč je zvýšen o jedna oproti tomuto maximu. Pokud je tabulka prázdná, je vygenerovanán klíč o hodnotě 1. Toto je provedeno procedurou id_generator.
    - Trigger je testován vložením záznamu s hodnotou primárního klíče NULL.
- dodat_zbozi
    - Po vložení nového záznamu do tabulky Dodavka, je hodnota Mnozstvi_zbozi v příslušném záznamu tabulky Pobocka inkrementována.

Uložené procedury:

- zaloha_dodavek_bez_ibalginu
    - Pro účely této procedury byla vytvořena tabulka Dodavka_zaloha, která má stený tvar jako tabulka Dodavka.
    - Tato procedura zkopíruje všechny záznamy v tabulce Dodavka, které nejsou dodávky ibalginu, do tabulky Dodavka_zaloha.
    - Pokud je v momentálním záznamu informace, že se jedná o dodávku ibalginu, je vyvolána uživatelská vyjímka, které tento záznam přeskočí.
- glob_zmena_cen
    - Parametr modifikator(NUMBER)
    - Změní cenu všech léků v databázi podle vzorce nova_cena=(cena/modifikator)*10
    - Pokud je jako modifikator zadána nula, je vyvolána vyjímka ZERO_DIVIDE a funkce je ukončena.

[1] https://cs.wikipedia.org/wiki/Rodn%C3%A9_%C4%8D%C3%ADslo
[2]https://cs.wikipedia.org/wiki/Identifika%C4%8Dn%C3%AD_%C4%8D%C3%ADslo_osoby#Struktura_I.C4.8CO

Index a EXPLAIN PLAN:

- Dotaz je: Kolik rezervací jakých léků je na pobočce s adresou Ostrava?
- Neoptimalizovaný dotaz bez explicitních indexů a s použitím spojení JOIN:

```
-------------------------------------------------------------------------------------------
| Id  | Operation                      | Name          | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------------------
|   0 |  SELECT STATEMENT              |               |    3 |    84 |    11  (28)| 00:00:01 |
|   1 |  SORT ORDER BY                 |               |    3 |    84 |    11  (28)| 00:00:01 |
|   2 |  HASH GROUP BY                 |               |    3 |    84 |    11  (28)| 00:00:01 |
|*  3 |  HASH JOIN                     |               |    4 |   112 |     9  (12)| 00:00:01 |
|   4 |  MERGE JOIN                    |               |    4 |    64 |     6  (17)| 00:00:01 |
|*  5 |  TABLE ACCESS BY INDEX ROWID   | POBOCKA       |    1 |    10 |     2   (0)| 00:00:01 |
|   6 |  INDEX FULL SCAN               | PK_POBOCKA |    3 |       |     1   (0)| 00:00:01 |
|*  7 |  SORT JOIN                     |               |   12 |    72 |     4  (25)| 00:00:01 |
|   8 |  TABLE ACCESS FULL             | REZERVACE     |   12 |    72 |     3   (0)| 00:00:01 |
|   9 |  TABLE ACCESS FULL             | LEK           |    3 |    36 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

  3 - access("R"."ID_LEKU"="L"."CISELNY_KOD")
  5 - filter("P"."ADRESA"='Ostrava')
  7 - access("P"."ID_POBOCKY"="R"."ID_POBOCKY")
      filter("P"."ID_POBOCKY"="R"."ID_POBOCKY")
```

    - Seřazení řádků
    - Shluknutí požadovaných duplicitních řádků.
    - Párovaní záznamů přes hash klíče – zahashování klíčů menší tabulky a následné postupné hashování a porovnávání řádků větší tabulky.
    - Spojení tabulek podle shodných klíčů.
    - Dohledání po průchodu indexem
    - Procházení řádků bez použitelného indexu – filtrace Poboček mimo Ostravu
    - Spojení tabulek porovnáváním seřazených klíčů.
    - Nakonec dojde ke kompletnímu prohledání tabulek Rezervace a Lek
- Přestože nebyl zvolen optimální typ spojení a filtrace pomocí where byla napsána až za spojení, proces byl automaticky optimalizován.
- Možným vylepšením je vytvoření indexu pro sloupec Nazev tabulky Lek, která byla celá procházena kvůli SELECTu. Je očekávatelné, že bude sloupec Nazev prohledáván často, je tedy dobrým kandidátem pro vytvoření indexu. Dalším vhodným kandidátem je sloupec Adresa tabulky Pobocka.

- EXPLAIN PLAN při použití indexů Lek(nazev) a Pobocka(Adresa):

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                  |     3 |    84 |     9  (23)| 00:00:01 |
|   1 |  SORT ORDER BY                 |                  |     3 |    84 |     9  (23)| 00:00:01 |
|   2 |   HASH GROUP BY                |                  |     3 |    84 |     9  (23)| 00:00:01 |
|*  3 |    HASH JOIN                   |                  |     4 |   112 |     7   (0)| 00:00:01 |
|*  4 |     HASH JOIN                  |                  |     4 |    64 |     5   (0)| 00:00:01 |
|   5 |      TABLE ACC … ROWID BATCHED | POBOCKA          |     1 |    10 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN         | POBOCKA_ADRESA   |     1 |       |     1   (0)| 00:00:01
|   7 |      TABLE ACCESS FULL         | REZERVACE        |    12 |    72 |     3   (0)| 00:00:01 |
|   8 |     VIEW                       | index$_join$_004 |     3 |    36 |     2   (0)| 00:00:01
|*  9 |      HASH JOIN                 |                  |       |       |            |          |
|  10 |       INDEX FAST FULL SCAN     | LEK_NAZEV        |     3 |    36 |     1   (0)| 00:00:01 |
|  11 |       INDEX FAST FULL SCAN     | PK_LEK           |     3 |    36 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("R"."ID_LEKU"="L"."CISELNY_KOD")
   4 - access("P"."ID_POBOCKY"="R"."ID_POBOCKY")
   6 - access("P"."ADRESA"='Ostrava')
   9 - access(ROWID=ROWID)
```

    - Zde je na první pohled vidět snížení ceny.
    - Od kroku 4:
    - Hash join místo Merge join – prakticky bez rozdílu v ceně
    - Table access rowid batched – změněný způsob získávání indexů, nový ve verzi oracle 12c – může snížit počet čtení
    - Díky indexu je provedena cílená filtrace podle adresy.
    - Výpis hodnot z indexovaných sloupců.
    - Poslední sjednocení s tabulkou Lek
    - Cílený výpis hodnot ze sloupců indexu.
- Cena byla i při malém množství dat v databázi značně snížena indexováním. Kupříkladu cena výpisu spadla na třetinu původní hodnoty.

Udělení práv:

- Uživateli xstude23 byla udělena práva provedení příkazu SELECT nad tabulkami Dodavka, Pobocka, Rezervace, Ma_v_sortimentu, Prispela, Na_predpis, Pojistovna a Lek.
- Tato práva byla úspěšně otestována příkazem: ALTER SESSION SET CURRENT_SCHEMA = xzaple36

Materializovaný pohled:

- Byl vytvořen materializovaný pohled pro dotaz:
    - SELECT P.ID_pobocky, P.Adresa, R.ID_rezervace, R.Jmeno_zakaznika, R.Datum, R.ID_leku FROM Pobocka P JOIN Rezervace R ON P.ID_pobocky = R.ID_pobocky;