
# IDS - Databázové systémy

## Databáze pro sociální síť

Aleš Sedláček (xsedla1c)

Jan Demel (xdemel01)

## Úvod

Skript nejprve vyčistí všechny předešlé vytvořené schémata a data. Následně vytvoří základní schéma databáze, naplní tabulky ukázkovými daty, přidělí práva druhému uživateli, vytvoří dvě procedury, triggery, vykoná sekvenci příkazů a vyzkouší příkaz EXPLAIN PLAN bez a s indexem za účelem porovnání rychlosti dotazování.

## Triggery

První trigger použitý v naší implementaci se jmenuje UZIVATEL_BEFORE_INSERT a má za úkol zabezpečit přidávání automatického unikátního ID pro nově vytvořeného uživatele, pokud není explicitně zadán v příkazu.

Druhý trigger má název ALBUM_BEFORE_DELETE a kompletuje doprovodnou činnost při mazání alba. Je totiž nutné před samotným příkazem DELETE album smazat i všechny fotky, které jsou v daném albu.

Poslední trigger má název FOTKA_BEFORE_DELETE a navazuje na předchozí funkcionalitu. Na fotce může být označen uživatel a tudíž se na každé fotce, která bude smazána, musí smazat i veškeré označení.

## Procedury

Naše procedura how_many_people_lives_at_address  počítá, kolik záznamů o uživatelích žijících na dané adrese existuje. Byl zde využit CURSOR a ošetřeny možné vyjímky. Rovněž byla vytvořena proměnná, pro jejíž deklaraci bylo využito %ROWTYPE.

## Explain plan + Index

Byl vytvořen dotaz, který má za úkol zjistit, kolik zpráv v jednotlivých konverzacích jednotliví uživatelé napsali. Je využito spojování tabulek pomocí INNER JOIN, klauzule GROUP BY pro sloučení výsledků podle zadaných parametrů a agregační funkce COUNT(). Nad tímto dotazem byl zavolán příkaz EXPLAIN PLAN, abychom zjistili detailní průběh procesu. Poté jsme vytvořili index a zkusili, zda se dotaz zrychlí.

**Před** přidáním indexu

```
Plan hash value: 3478564947

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     1 |   133 |     8  (13)| 00:00:01 |
|   1 |  HASH GROUP BY                |               |     1 |   133 |     8  (13)| 00:00:01 |
|   2 |   NESTED LOOPS                |               |     1 |   133 |     7   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |               |     1 |   133 |     7   (0)| 00:00:01 |
|*  4 |     HASH JOIN                 |               |     1 |    93 |     6   (0)| 00:00:01 |
|*  5 |      TABLE ACCESS FULL        | UZIVATEL      |     1 |    67 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL        | ZPRAVA        |    11 |   286 |     3   (0)| 00:00:01 |
|*  7 |     INDEX UNIQUE SCAN         | KONVERZACE_PK |     1 |       |     0   (0)| 00:00:01 |
|   8 |    TABLE ACCESS BY INDEX ROWID| KONVERZACE    |     1 |    40 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

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

   4 - access("ZPRAVA"."UZIVATEL"="UZIVATEL"."ID")
   5 - filter("UZIVATEL"."PRIJMENI"='Sedláček')
   7 - access("KONVERZACE"."ID"="ZPRAVA"."KONVERZACE")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan
```

**Po** přidání indexu

```
Plan hash value: 3074327151

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |               |     1 |   133 |     7  (15)| 00:00:01 |
|   1 |  HASH GROUP BY                           |               |     1 |   133 |     7  (15)| 00:00:01 |
|   2 |   NESTED LOOPS                           |               |     1 |   133 |     6   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                          |               |     1 |   133 |     6   (0)| 00:00:01 |
|*  4 |     HASH JOIN                            |               |     1 |    93 |     5   (0)| 00:00:01 |
|   5 |      TABLE ACCESS BY INDEX ROWID BATCHED| UZIVATEL      |     1 |    67 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN                   | TEST_INDEX1   |     1 |       |     1   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL                   | ZPRAVA        |    11 |   286 |     3   (0)| 00:00:01 |
|*  8 |     INDEX UNIQUE SCAN                    | KONVERZACE_PK |     1 |       |     0   (0)| 00:00:01 |
|   9 |    TABLE ACCESS BY INDEX ROWID           | KONVERZACE    |     1 |    40 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------------

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

   4 - access("ZPRAVA"."UZIVATEL"="UZIVATEL"."ID")
   6 - access("UZIVATEL"."PRIJMENI"='Sedláček')
   8 - access("KONVERZACE"."ID"="ZPRAVA"."KONVERZACE")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan
```

Jde vidět menší vytížení CPU díky správnému indexování tabulky uzivatel a jejího atributu příjmení.

## Materializovaný pohled

Simuluje vytváření akce, která se uveřejní pro všechny uživatele až tehdy, pokud dojde ke commitu. Tudíž se změny při insertování do tabulky změní jen lokálně u jednoho uživatele, dokud sám neuveřejní změny pro všechny.

## ER Diagram