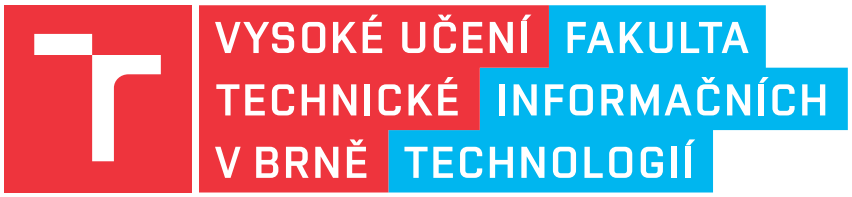
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Databázové systémy 2018/2019

Zadanie č. 8 – Autoopravna

1. mája 2019

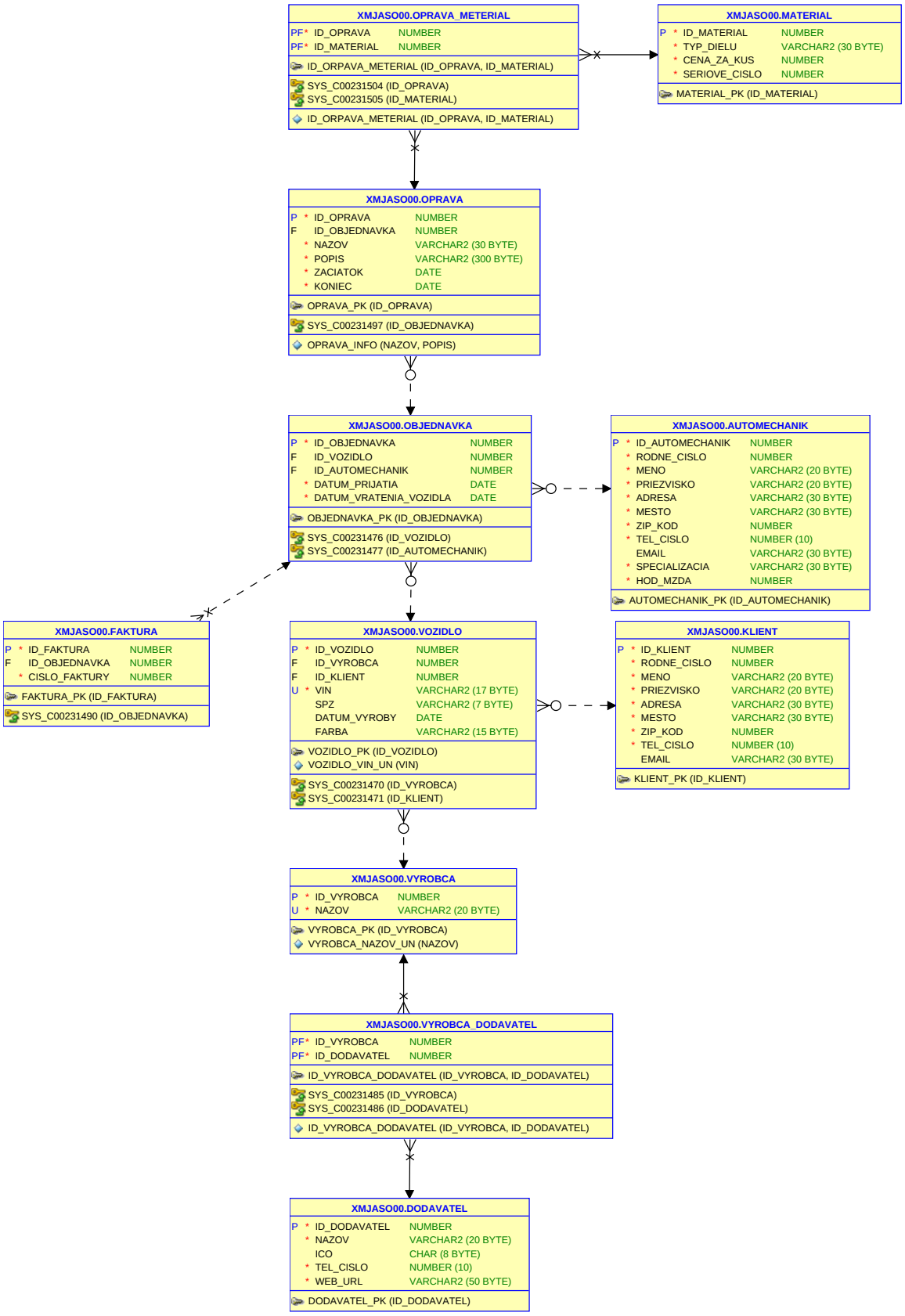
Igor Mjasojedov **(xmjaso00)**
Alex Sporni (xsporn01)

Obsah

1 Zadanie

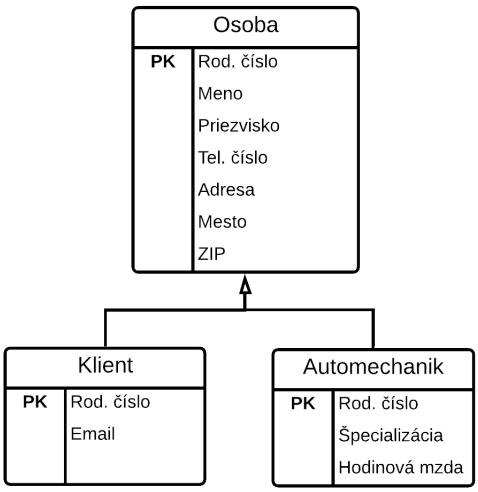
Navrhněte IS autoopravny ke kterému mají přístup pouze zaměstnanci dílny. Systém má umožnit přijímat objednávky, přiřazovat je automechanikům specializovaným na určitý typ oprav a plánovat termín oprav v závislosti na vytížení automechaniků. Objednávka by měla obsahovat údaje o vozidle (barva, datum výroby, autorádio, značka auta, SPZ). U značky auta je třeba uchovávat informace o kontaktu na dodavatele náhradních dílů. Dále objednávka obsahuje datum a čas převzetí auta a předpokládané datum dodání opraveného auta. U jednotlivých typů oprav je informace o ceně, která se může skládat z ceny použitého materiálu a doby strávené automechanikem na opravě. Každý automechanik má dánu hodinovou mzdu. Systém umožňuje také evidovat množství materiálu pro opravy aut. Faktura vystavená zákazníkovi obsahuje jednotlivé výpisy použitého materiálu a času stráveného na opravě jednotlivými automechaniky.

2 Schéma relačnej databázy



3 Generalizácia/špecializácia

Generalizáciu/špecializáciu sme v našom zadaní využili pri entite `Osoba`, kde osoba môže byť typu klient alebo automechanik. Daný problém sme vyriešili pomocou dvoch tabuliek pre podtypty `Klient` a `Automechanik` i s atribútnmi nadtypu.



4 Implementácia

Skript najprv zahodí pomocou príkazu `DROP` všetky vytvorené tabuľky, sekvencie, indexy a pohľady aby sa predišlo možným konflikom. Skript následne vytvorí základné objekty schémy databázy t.j. tabuľky, integritné obmedzenia (PK, FK). Skript následne naplní vytvorené tabuľky testovacími dátami, nad ktorými sa prevedie niekoľko príkazov `SELECT`, ktoré pokrývajú požiadavky projektu.

4.1 Triggery

Naša implementácia obsahuje celkovo 3 triggery, pričom sa všetky spúšťajú pred vložením testovacích hodnôt do tabuľky na čo sme využili príkazy `BEFORE INSERT ON` a `BEFORE INSERT OR UPDATE ON`

Prvý netriviálny databázový trigger, ktorý explicitne vyplýva zo zadania slúži pre automatické generovanie hodnôt primárneho kľúča pre tabuľku `klient`.

Druhý netriviálny databázový trigger slúži na validáciu rodného čísla klienta. Trigger pozostáva z viacerých kontrol. Prvá kontroluje či je rodné číslo deliteľné číslom 11 a ak áno, tak potom nasleduje kontrola či sa jedná o ženu. V prípade že na pozícii mesiaca v rodnom čísle sa nachádza číslo vačšie ako 50, v tom prípade sa od daného čísla odčíta táto hodnota. Ak sa jedná o validnú hodnotu mesiaca z rozsahu 1-12 tak predpokladáme správnosť výsledku. Posledná kontrola spočíva v overení či na pozícii dňa v rodnom čísle sa nenachádza číslo väčšie ako 31.

Tretí databázový trigger slúži na kontrolu dátumu. Jedná sa o porovnanie začiatku a konca opravy. V prípade, že počiatočný dátum je neskorší ako koncový dátum opravy, jedná sa o nevalidný dátum opravy. V prípade akejkoľvej chyby sa na štandardný chybový výstup vypíše prislúchajúce chybové hlásenie.

Na konci skriptu sme demonštrovali použitie triggerov, a to tak, že do tabuliek sme vložili nevalidné hodnoty. Pri teste prvého triggeru sme vložili nevalidné rodné číslo a pri druhom teste sme vložili nevalidný formát dátumu, kde dátum konca opravy predchádza dátum začiatku opravy vozidla.

4.2 Procedúry

Projekt obsahuje celkovo 2 netriviálne procedúry. Obe procedúry obsahuju použitie kurzora. Obe procedúry obsahujú premenné s dátovým typom odkazujúcim sa na riadok tabuľky `table_name%ROWTYPE`.
Procedúra `prevadzka_zataz()` nám slúži na vypísanie záťaže autoopravovne v zadanom meste na základe vyťaženosti jednotlivých automechanikov. Pre potreby implementácie sa vytvoril cursor implementujúci nasledujúci select: `SELECT * FROM automechanik natural left join objednavka natural left join oprava;`. Daným selctom získame informácie o automechanikoch a objednávkach s opravami na ktorých pracujú. Pri získaní aktuálneho dátumu a počtu automechanikov v danej prevádzke sa prostredníctvom cyklu `LOOP` spracovali jednotlivé záznamy z kurzora. Počas vykonávania cyklu sa porovná aktuálny dátum s dátumom špecifikovaného ako plánovaného konca opravy. V prípade, že ukončenie orpavy je iba v pláne počet pracujúcich automechanikov na oprave sa zvyšuje. Následne sa vypočíta percentuálna vyťaženosť a vypíše sa informácia o prevádzke, vyťaženosti a počte automechanikov na prevádzke. V prípade nulového počtu automechanikov na prevádzke by počet zaťaženosti nebol možný `ZERO_DIVIDE` a túto situáciu sme vyriešili zachytením výnimky, pri ktorej sa vypísala prisláchajúca chybová hláška.

Procedúra `aktivne_opravy()` Zmyslom tejto procedúry je vypísať všetky opravy, na ktorých sa aktuálne pracuje. Tak ako aj v predošlej procedúre, po zistení aktuálneho dátumu sa prostredníctvom cyklu vyhodnotí každý záznam v kurzore a v prípade vykonávajúcej sa práve opravy, prislúchajúce informácie sú vypísané na výstup. Po implementácií oboch procedúr sme vyskúšali ich funkčnosť a ukážkové volanie funkcií s ich výstupom je uvedené nižšie:

EXECUTE prevadzka_zataz('Nove Zamky')

```
Prevadzka: Nove Zamky | Vytazenost: 50% | Pocet automechanikov: 2
```

EXECUTE aktivne_opravy()

```
OPRAVA: Stahovanie okien| POPIS: poskodenie motorceka z dovodu zapadnutia prachom
OPRAVA: Brzdovy system| POPIS: poskodene brzdy na zadnej naprave = vydraty kotuc
OPRAVA: Elektroinstalacia| POPIS: nefunkcna palubna doska
OPRAVA: Vymena oleja| POPIS: Servisna prehliadka-vymena oleja po 15000km
```

4.3 Explain plan a vytvorenie indexu

Explain plan nám slúži na zobrazenie postupnosti realizácie operácií optimalizátorom Oracle pre dotaz `SELECT`, ktorým získame názov opravy, popis opravy, vin číslo vozidla a sumu náhradných dielov. Do úvahy sa berú iba opravy, ktoré sa začali vykonávať v priebehu Júna roku 2018 a ktoré majú sumu cien dielov väčšiu ako 150.

```
SELECT O.nazov, O.popis, V.vin, SUM(M.cena_za_kus) AS naklady
FROM oprava_meterial NATURAL JOIN material M NATURAL JOIN oprava O NATURAL JOIN objednavka NATURAL JOIN vozidlo V
WHERE zaciatok BETWEEN '01-06-2018' AND '30-06-2018'
GROUP BY O.nazov, O.popis, V.vin
HAVING SUM(M.cena_za_kus) > 150
ORDER BY naklady;
```

Ukážka výstupu bez použitia indexu

Použitie indexu môže byť praktické v prípade častého vyhľadávania v určitej tabuľke. Nevýhoda nastáva v prípade častého editovania tabuľky, index je potrebné taktiež aktualizovať, čo má zá následok spomalenie prístupu. V našom prípade sme vytvorili index `INDEX oprava_info_I`

```
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                    |     4 |  1220 |    16  (13)| 00:00:01 |
|   1 |  SORT ORDER BY                      |                    |     4 |  1220 |    16  (13)| 00:00:01 |
|*  2 |   FILTER                            |                    |       |       |            |          |
|   3 |    HASH GROUP BY                    |                    |     4 |  1220 |    16  (13)| 00:00:01 |
|*  4 |     HASH JOIN                       |                    |     4 |  1220 |    14   (0)| 00:00:01 |
|*  5 |      HASH JOIN                      |                    |     4 |  1116 |    11   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
|   6 |       NESTED LOOPS                  |                    |     3 |   759 |     9   (0)| 00:00:01 |
|   7 |        NESTED LOOPS                 |                    |     3 |   759 |     9   (0)| 00:00:01 |
|   8 |         NESTED LOOPS                |                    |     3 |   690 |     6   (0)| 00:00:01 |
|*  9 |          TABLE ACCESS FULL          | OPRAVA             |     3 |   612 |     3   (0)| 00:00:01 |
|  10 |          TABLE ACCESS BY INDEX ROWID| OBJEDNAVKA         |     1 |    26 |     1   (0)| 00:00:01 |
|* 11 |           INDEX UNIQUE SCAN         | SYS_C00247441      |     1 |       |     0   (0)| 00:00:01 |
|* 12 |         INDEX UNIQUE SCAN           | SYS_C00247430      |     1 |       |     0   (0)| 00:00:01 |
|  13 |        TABLE ACCESS BY INDEX ROWID  | VOZIDLO            |     1 |    23 |     1   (0)| 00:00:01 |
|  14 |       INDEX FAST FULL SCAN          | ID_ORPAVA_METERIAL |     6 |   156 |     2   (0)| 00:00:01 |
|  15 |      TABLE ACCESS FULL              | MATERIAL           |     9 |   234 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```

Ukážka výstupu s použitím indexu

```
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                    |     4 |  1220 |    15  (14)| 00:00:01 |
|   1 |  SORT ORDER BY                      |                    |     4 |  1220 |    15  (14)| 00:00:01 |
|*  2 |   FILTER                            |                    |       |       |            |          |
|   3 |    HASH GROUP BY                    |                    |     4 |  1220 |    15  (14)| 00:00:01 |
|*  4 |     HASH JOIN                       |                    |     4 |  1220 |    13   (0)| 00:00:01 |
|*  5 |      HASH JOIN                      |                    |     4 |  1116 |    10   (0)| 00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
|   6 |       NESTED LOOPS                  |                    |     3 |   759 |     8   (0)| 00:00:01 |
|   7 |        NESTED LOOPS                 |                    |     3 |   759 |     8   (0)| 00:00:01 |
|   8 |         NESTED LOOPS                |                    |     3 |   690 |     5   (0)| 00:00:01 |
|   9 |          TABLE ACCESS BY INDEX ROWID BATCHED| OPRAVA             |     3 |   612 |     2   (0)| 00:00:01 |
|* 10 |           INDEX RANGE SCAN          | OPRAVA_INFO_I      |     3 |       |     1   (0)| 00:00:01 |
|  11 |          TABLE ACCESS BY INDEX ROWID| OBJEDNAVKA         |     1 |    26 |     1   (0)| 00:00:01 |
|* 12 |           INDEX UNIQUE SCAN         | SYS_C00247441      |     1 |       |     0   (0)| 00:00:01 |
|* 13 |         INDEX UNIQUE SCAN           | SYS_C00247430      |     1 |       |     0   (0)| 00:00:01 |
|  14 |        TABLE ACCESS BY INDEX ROWID  | VOZIDLO            |     1 |    23 |     1   (0)| 00:00:01 |
|  15 |       INDEX FAST FULL SCAN          | ID_ORPAVA_METERIAL |     6 |   156 |     2   (0)| 00:00:01 |
|  16 |      TABLE ACCESS FULL              | MATERIAL           |     9 |   234 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------
```

Ako je možné vidieť na danom výstupe, výkonnosť prevedenia dotazu(`cost`) sa zvýšila resp. znížila. V prípade väčšieho množstva dát by bol rozdiel medzi jednotlivými výstupmi viditelnejší.

4.4 Prístupové práva

Prístupové práva v kontexte našej témy simulujeme vytvorením práv pre administratívneho pracovníka a automechanika. Administratívny pracovník prirodzene disponuje všetkými oprávneniami na rozdiel od automechanika, ktorý sa z logického hľadiska stretáva s niekoľkými obmedzeniami pri prístupe k vybraným tabuľkám. Znamená to, že nemôže použiť procedúru `prevadzka_zataz()`, ktorá slúži na výpis percentuálnej záťaže vyťaženosti automechanikov v danom meste. Rovnako nemôže vkladať záznamy do tabuliek, okrem tabuliek `oprava`, `material`,a procedúry `aktivne_opravy`, ktoré potrebuje k náplni práce.

4.5 Materializovaný pohľad

V projekte sme implementovali materializovaný pohľad patriaci druhému členovi tímu. Materializovaný pohľad slúži na uloženie často využívaného pohľadu lokálne na disk, z dôvodu rýchlejšieho prístupu pri opakovanom žiadaní o daný pohľad. V našej implementácií sme v provom rade vytvorili materializované logy, ktoré sme následne využívali pri materialozovaných pohľadoch, konkrétne pri `REFRESH FAST ON COMMIT`. Taktiež sme využili možnosť cacheovania prostredníctvom možnosti `cache` a select, ktorý sme týmto pohľadom implementovali je nasledujúci:

```
SELECT id_automechanik, COUNT(id_automechanik) AS Priradene_Objednavky
FROM objednavka NATURAL JOIN automechanik
GROUP BY id_automechanik;
```

A ako verifikáciu funkčnosti nášho pohľadu porovali výstupy klasického pohľadu a materializovaného pohľadu po vložení záznamu do tabuľky objendnávka.

5 Záver

Skript sme riadne vypracovali a otestovali v nástroji *Oracle SQL Developer* v prostredí *Oracle* na školskom serveri *Oracle 12c*. K úspešnému vypracovaniu projektu nám pomohli vedomosti nadobudnuté z predmetu IDS a oficiálnej dokumentácie *Oracle*. Na záver by sme chceli kladne ohodnotiť aj možnosť demonštačných cvičení, ktoré nám často pomohli pri konkrétnej problematike.