VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií

DATABASE SYSTEMS
2018/2019

Dokumentace k projektu

Zadání č. 27 – Hotel

Radim Šustek (xsuste11) Brno, 29. duben 2019

Úvod
Skript obsahuje základní implementaci schématu databáze včetně 2 databázových triggerů a 2 procedur včetně testovacích dat.

TRIGGER
Implementace obsahuje 2 databázové triggery. Databázové triggery se implementují ještě před vložením testovacích dat.

1) ***Navyseni_HostID*** slouží pro automatické navyšování primárního klíče (PK) v tabulce Hosts. Hodnoty jsou získávány ze sekvence ***HostID_sekvence***. Nový primární klíč se vygeneruje, pokud je primární klíč null.

2) ***Kontrola_vypoctu_ceny*** tento trigger hlídá, že vypočtená cena v tabulce Payment je skutečně správná a že odpovídají jednotlivé podsložky platby.

PROCEDURY
Skript obsahuje 2 procedury. Jednu bez parametrů a druhou s parametrem jméno. Obě procedury vypisují výsledek na DBMS_OUTPUT a využívají
(table_name.column_name%TYPE)

1) ***Obsazenost_hotelu*** tato procedura nevyžaduje žádný parametr. Procedura vyhodnotí procentuální obsazení hotelu. V případě že žádný z pokojů není obsazen je vyvolána výjimka ***zero_divide***, která na DBMS_OUTPUT vypíše informační zprávu.

2) ***Kdo_vyuzil_sluzbu*** procedura vypíše seznam všech uživatelů, kteří službu využily. Pro použití této procedury je třeba vložit jméno hledané služby.

MATERIALIZED VIEW
Materializovaný pohled lze využít tehdy, když nejsou potřeba aktuální data a chceme snížit zatížení databáze. Aby bylo tohle dokázáno, vytvořil jsem 2 pohledy ***pohled_hosts (PH)*** a ***pohled_hosts_materialization (PHM)*** na tabulku Hosts, přičemž pokud následně do Hosts vložíme novou položku, můžeme vidět, že pohled PH danou položku obsahuje, ale PHM ne.

PRÁVA
Práva pro přístup k jednotlivým tabulkám udělíme pomocí příkazu
GRANT ALL ON tablename to login;

EXPLAIN PLAN
Pro demonstraci jsme zvolili dotaz, který vyhledá pokoje s počtem uskutečněných pobytů více než jeden. Explain plan slouží ke zjištění průběhu provedení daného dotazu. Poprvé bez použití indexu, díky kterému lze optimalizovat zpracování dotazů a poté s indexem pro urychlení

Při použití bez indexu

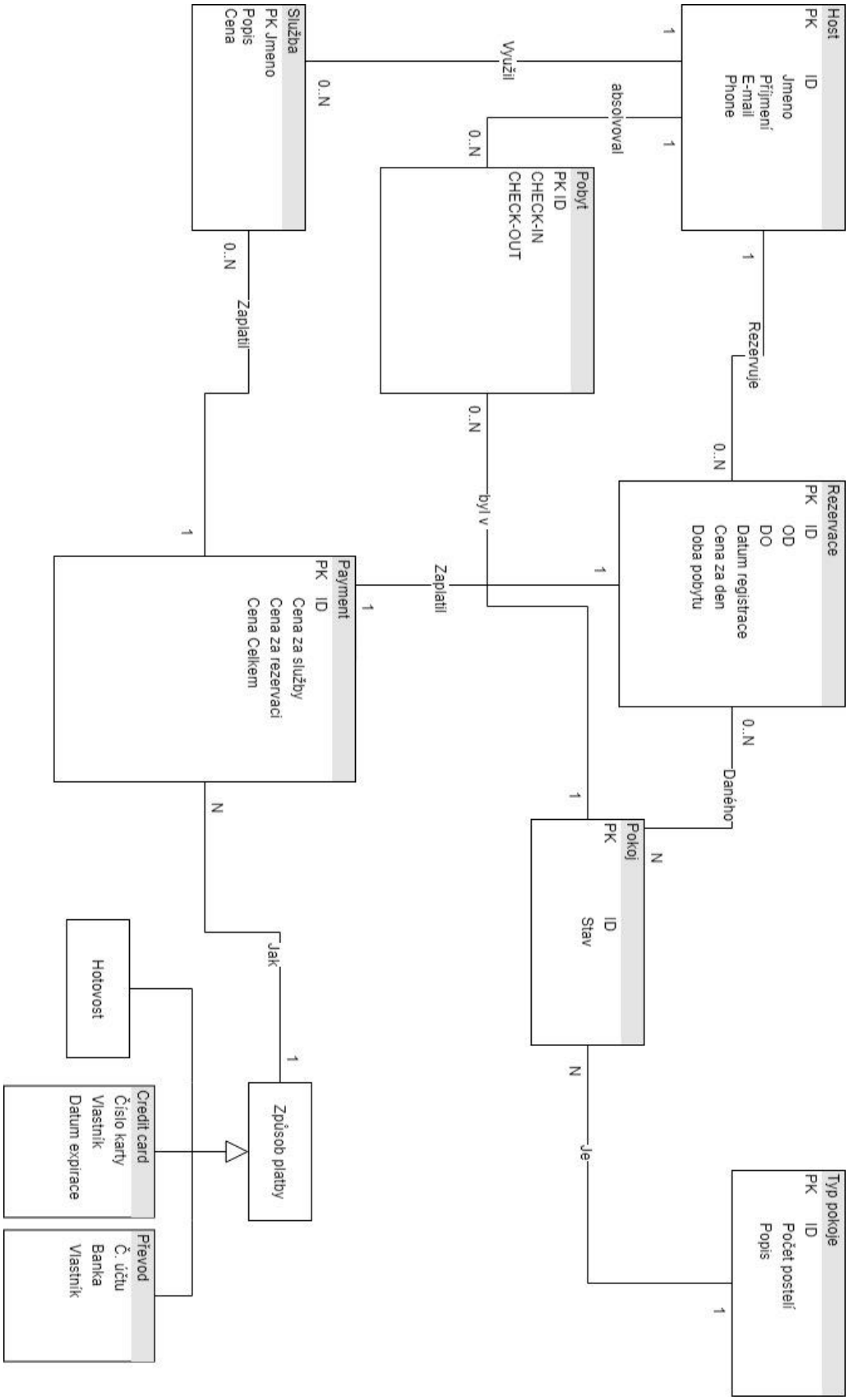
```
Plan hash value: 3779123003

-------------------------------------------------------------------------------
| Id  | Operation            | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |        |     4 |   104 |     7  (15)| 00:00:01 |
|*  1 |  FILTER              |        |       |       |            |          |
|   2 |   HASH GROUP BY      |        |     4 |   104 |     7  (15)| 00:00:01 |
|*  3 |    HASH JOIN         |        |     4 |   104 |     6   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| POBYT  |     4 |    52 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| POKOJE |     7 |    91 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter(COUNT(*)>1)
   3 - access("POBYT"."POKOJID"="P"."POKOJID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
```

S indexem

```
Plan hash value: 4237293593

-------------------------------------------------------------------------------
| Id  | Operation            | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |         |     4 |   104 |     4  (25)| 00:00:01 |
|*  1 |  FILTER              |         |       |       |            |          |
|   2 |   HASH GROUP BY      |         |     4 |   104 |     4  (25)| 00:00:01 |
|   3 |    NESTED LOOPS      |         |     4 |   104 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| POBYT   |     4 |    52 |     3   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN| EXP_IND |     1 |    13 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter(COUNT(*)>1)
   5 - access("POBYT"."POKOJID"="P"."POKOJID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
```

Zde můžete vidět, že při použití indexu je u daného dotazu menší jak využití paměti tak CPU.

ERD


Host
PK ID
Jmeno
Příjmení
E-mail
Phone
Služba
PK Jmeno
Popis
Cena
Využil
absolvoval
Pobyt
PK ID
CHECK-IN
CHECK-OUT
Rezervuje
Rezervace
PK ID
OD
DO
Datum registrace
Cena za den
Doba pobytu
Zaplatil
byl v
Zaplatil
Payment
PK ID
Cena za služby
Cena za rezervaci
Cena Celkem
Daného
Pokoj
PK ID
Stav
Je
Typ pokoje
PK ID
Počet postelí
Popis
Jak
Způsob platby
Hotovost
Credit card
Číslo karty
Vlastník
Datum expirace
Převod
Č. účtu
Banka
Vlastník