VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

Fakulta informačních technologií
Faculty of Information Technology

DATABÁZOVÉ SYSTÉMY

2016/2017

Zadání 34 - Ordinace praktického lékaře

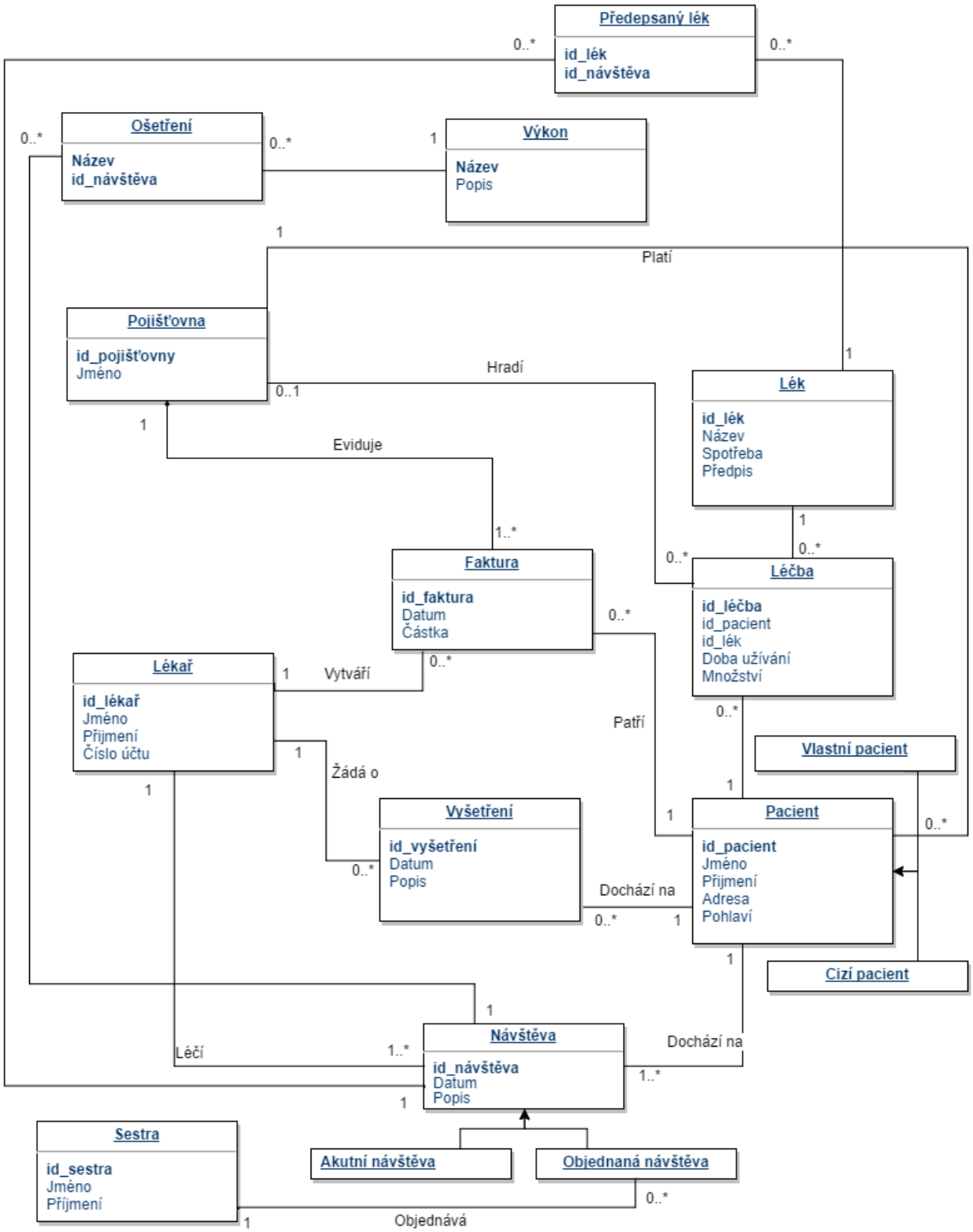
Jakub Svoboda (xsvobo0z)
Michael Schmid (xschmi08)

28.4.2017

Zadání:

V ordinaci praktického lékaře jsou ošetřováni objednaní pacienti i akutní případy cizích pacientů. Uvažujte ordinaci jednoho lékaře a jedné sestry. Systém umožní sestře objednávat pacienty na daný termín a případně i specifikovat jaké výkony jsou v plánu během tohoto termínu provést. Dále eviduje návštěvy pacienta (ať již byly předem naplánovány anebo šlo o akutní případ). U každé návštěvy je třeba také evidovat jaké výkony byly provedeny a jaké léky byly předepsány. Systém bude zvát pacienty na pravidelné prohlídky a očkování, které mají různou dobu účinnosti. Pacient je vždy pojištěncem jediné pojišťovny, předpokládejte, že nás nezajímají změny pojišťovny, které pacient v průběhu času provedl. Systém musí evidovat faktury pro pojišťovnu a umožnit lékaři je vytisknout. V průběhu jedné návštěvy mohl lékař zažádat o (jedno či více) vyšetření, na které musel pacient dojít na jiné pracoviště.

Schéma databáze:



Implementace:

Skript nejprve vytvoří tabulky a naplní je ukázkovými daty. Dále jsou vytvořeny dvě procedury a je provedeno jejich ukázkové volání. Následně je vytvořen index a explain plan, jsou nastavena přístupová práva pro druhého uživatele a je vytvořen materializovaný pohled.

Triggery:

Pro projekt jsou vytvořeny dva triggery, které se spouští při vložení nebo úpravě položek tabulky. První trigger kontroluje správnost mezinárodního bankovního čísla IBAN v tabulce lékař. Je požadováno číslo účtu v české bance, délka čísla tedy musí být 24 znaků a dva první znaky musí být „CZ“. Následně jsou první čtyři znaky převedeny na konec řetězce a všechna písmena jsou převedena na číslice. Zbytek po dělení výsledného čísla číslem 97 musí být roven jedné.

Druhý trigger se spouští při vložení nebo úpravě položek z tabulky pacient. Pokud je při vložení nového pacient zadána hodnota NULL pro sloupec id_pacient, je pacientovi vygenerována nová hodnota ze sekvence.

Procedury:

Procedura *pacient_prumerne_zaplatil(id_pacient NUMBER)* vyhledá všechny faktury, které pacient s daným ID zaplatil. Pomocí kurzoru jsou načtena data o pacientovi a ceně faktury. Pokud pro daného pacienta není v databázi zadána žádná faktura, došlo by při výpočtu průměrné hodnoty k dělení nulou. Toto vyvolá výjimku, která je v proceduře ošetřena.

Druhá procedura *detail_vykonu(id_vykon vykon.nazev%TYPE)* vyhledá v tabulce lékařských výkonů položku se shodným jménem a vypíše její podrobný popis. Pokud položka s daným názvem neexistuje, je vyvolána výjimka NO_DATA_FOUND, která je v proceduře ošetřena.

Explain plan a index

Použití explain plan je demonstrováno na select dotazu. Ten je nejprve spuštěn bez použití indexu, abychom mohli změřit náročnost a bylo zřejmé, jak databáze daný dotaz zpracovává. Poté je vytvořen index pro jména pacientů a dotaz je spuštěn znovu.

Bez použití indexu:

```
-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name            | Rows  | Bytes | Cost (%CPU)    | Time      |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                 |     1 |   119 |     6   (0)    | 00:00:01 |
|   1 |  SORT GROUP BY NOSORT          |                 |     1 |   119 |     6   (0)    | 00:00:01 |
|*  2 |   HASH JOIN                    |                 |     1 |   119 |     6   (0)    | 00:00:01 |
|*  3 |    TABLE ACCESS FULL           | PACIENT         |     1 |    80 |     3   (0)    | 00:00:01 |
|*  4 |    TABLE ACCESS FULL           | FAKTURA         |     7 |   273 |     3   (0)    | 00:00:01 |
-----------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("F"."ID_PACIENT"="P"."ID_PACIENT" AND
              "F"."ID_POJISTOVNA"="P"."ID_POJISTOVNA")
   3 - filter("P"."JMENO"='Jack' AND "P"."PRIJMENI"='ONeill')
   4 - filter("F"."CASTKA">50)
```

Po přidání indexu:

```
-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name            | Rows  | Bytes | Cost (%CPU)    | Time      |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                 |     1 |   119 |     5   (0)    | 00:00:01 |
|   1 |  SORT GROUP BY NOSORT          |                 |     1 |   119 |     5   (0)    | 00:00:01 |
|*  2 |   HASH JOIN                    |                 |     1 |   119 |     5   (0)    | 00:00:01 |
|*  3 |    TABLE ACCESS BY INDEX ROWID| PACIENT         |     1 |    80 |     2   (0)    | 00:00:01 |
|*  4 |     INDEX RANGE SCAN           | JMENO_INDEX     |     2 |       |     1   (0)    | 00:00:01 |
|*  5 |    TABLE ACCESS FULL           | FAKTURA         |     7 |   273 |     3   (0)    | 00:00:01 |
-----------------------------------------------------------------------------------------------------------------------
 Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("F"."ID_PACIENT"="P"."ID_PACIENT" AND
              "F"."ID_POJISTOVNA"="P"."ID_POJISTOVNA")
   3 - filter("P"."PRIJMENI"='ONeill')
   4 - access("P"."JMENO"='Jack')
   5 - filter("F"."CASTKA">50)
```

Při běhu s indexem se snížila položka Cost, tedy přístup na disk. Zátěž procesoru je v obou případech pod hranicí měřitelnosti.

Práva přístupu a materializovaný pohled

Druhému členovi týmu byla udělena práva pro přístup do tabulek a spouštění procedur *detaily_vykonu()* a *pacient_prumerne_zaplatil()*.

Dále byl vytvořen materializovaný pohled patřící druhému členovi, kdy byly nejprve vytvořeny materializované logy, aby při změně nemusela být tabulka aktualizována, ale uchovávaly se pouze změny. Poté byl již vytvořen samotný pohled. Při jeho vytváření jsme využili několika způsobů optimalizace:

Nologging:	zamezí zaznamenávání operací s pohledem.
Cache:	často používané položky jsou uloženy do cache
Build immediate:	pohled se naplní hned po vytvoření.
Enable query rewrite:	pohled bude použitý v optimalizátoru.