VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačných technológií

Databázové systémy

2018/2019

Zadanie - Veterinární klinika

Ondrej Šajdík (xsajdi01)

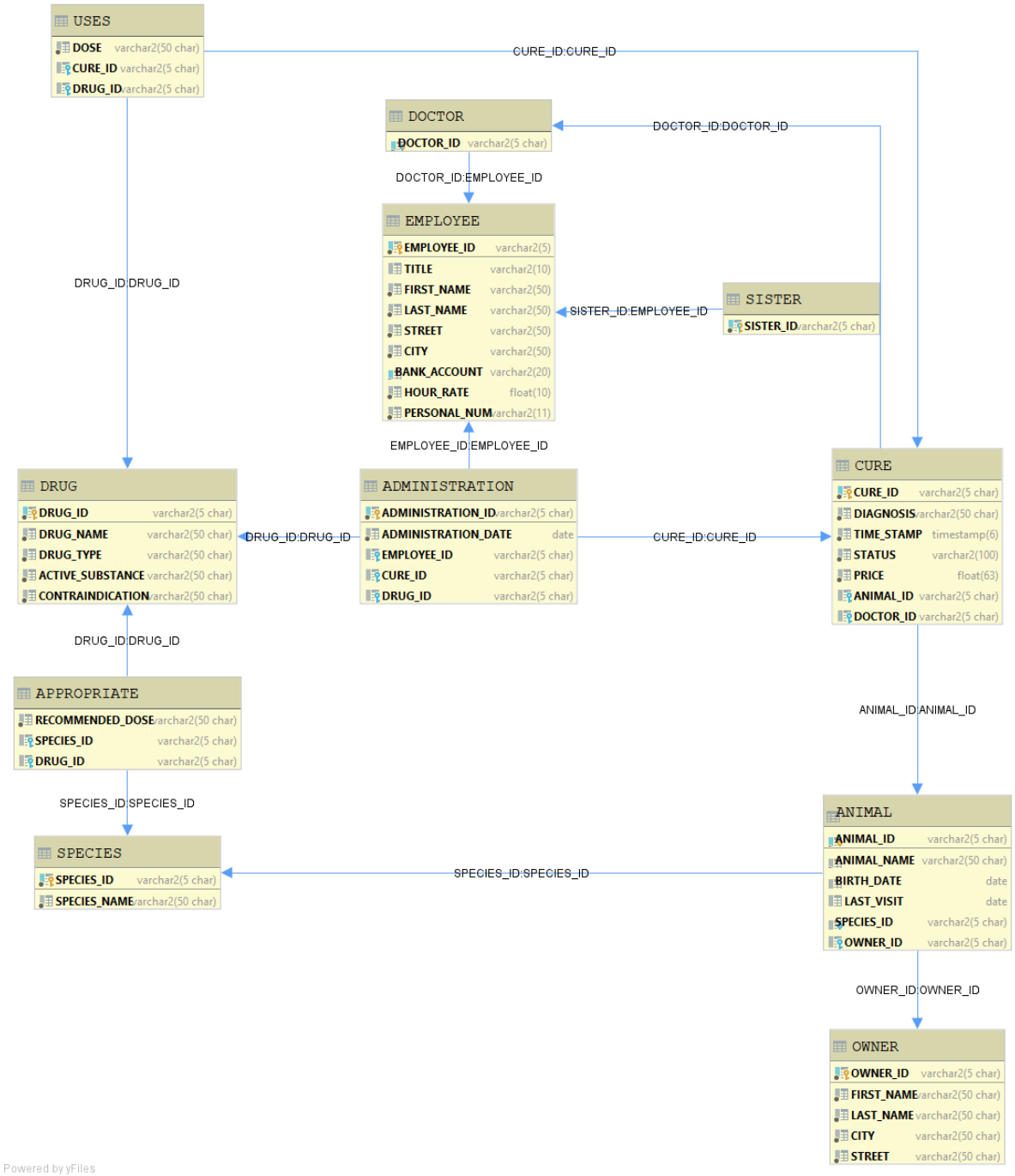
Tomáš Hampl (xhampl10)

29.4.2019

1. Zadanie

Namodelujte jednoduchý informační systém veterinární kliniky, kde se střídá více lékařů veterinářů a sester. O sestrách a veterinářích informační systém uchovává osobní informace jako jméno, titul, adresa, číslo účtu, hodinová mzda, přičemž lékaři mají přístup k údajům sester, ale sestry k údajům lékařů nikoliv. Hlavním účelem informačního systému je správa informací o léčených zvířatech, jednotlivých léčbách a jejich majitelích. U každého zvířete, které podstoupilo alespoň jednu léčbu, je v systému vytvořen záznam (o léčbě i o majiteli zvířete). Majitele zvířete lze v systému vyhledat podle jména, příjmení či adresy. Jeden majitel může samozřejmě vlastnit více domácích mazlíčků, o kterých je vytvořen záznam s jejich jménem, o jaký druh zvířete se jedná (kočka, pes, křeček, atd.), datum narození, datum poslední prohlídky a informace o jednotlivých léčbách, které dané zvíře podstoupilo (diagnóza, datum zahájení léčby, stav, cena). Při léčbě mohou být naordinovány léky s určitým dávkováním a pro specifikovanou dobu podávání. U léků předpokládejte možnost dohledat typ léku, jeho účinnou látku, kontraindikace, pro jaký druh zvířete se hodí (kočka, pes, ...) a doporučené dávkování pro daný druh zvířete. Jeden typ léku se může hodit pro více druhů zvířat s různým dávkováním a pro léčbu různých nemocí (nemoc modelovat nemusíte, postačí pouze název). V případě podání léku přímo v ordinaci (např. injekce) je k dispozici informace, kdo daný lék podal, stejně tak u léčby pro konkrétní zvíře lze dohledat, který veterinář ji vypsal.

2. Finální schéma databáze



2.1. Generalizácia

Generalizáciu sme použili u našej databáze pri tabuľke zamestnanec. Rozdelili sme ich na doktorov a sestry, pretože len doktor môže stanoviť liečby.

3. Implementácia

3.1. Triggery

V projektu sú vytvorené dva triggery, ktoré sa spúšťajú pri vložení alebo úprave dát.

Prví trigger, *owner_id_inc*, sa spúšťa pri vkladaní dát do tabuľky *owner*. Pokiaľ je při vložení nastavená hodnota *owner_id* na NULL, dôjde ku vygenerovaní hodnoty zo sekvencie *owner_seq*.

Druhý trigger, *personal_num*, sa spustí pri vkladaní nebo aktualizácii dát tabuľky *employee*, kde kontroluje správnosť rodného čísla *personal_num.* Pro správne vloženie je požadované dodržať typ (YYMMDD/XXXX). Ďalej sa kontroluje správnosť zadaného dátumu.

3.2. Procedúry

Implementovali sme dve procedúry *speciesCount* a *titles*, ktoré využívajú kurzory kvôli práci s viacerými riadkami databáze. Taktiež je v oboch použitá premenná s dátovým typom odkazujúcim sa na riadok tabuľky (table_name%ROWTYPE). A v prípade procedury *speciesCount*, aj premenná odkazujúca sa na typ stĺpca tabuľky (table_name.column_name%TYPE).

Procedúra speciesCount vypíše meno druhu, počet liečených zvierat daného druhu a ako veľké percento to predstavuje zo všetkých zvierat. Procedúra berie jeden argument, id daného druhu. Pri zlyhaní sa vypíše hláška s kódom -20001.

Procedúra *titles* vypíše pre každý titul v databázy počet zamestnancov s daným titulom a ich priemerný plat. Pri zlyhaní sa vypíše hláška s kódom -20002.

3.3. Explain plan a vytvorenie indexu

Explain plan sme demonštrovali na jednoduchom SELECT dotaze:

```
SELECT o.first_name, o.last_name, count(a.owner_id)
FROM owner o, animal a WHERE a.owner_id=o.owner_id
GROUP BY o.first_name, o.last_name;
```

Spustili sme explain plan **bez použitia indexu** a výsledkom bolo:

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |     5 |  1140 |     4  (25)| 00:00:01 |
|   1 |  HASH GROUP BY                |           |     5 |  1140 |     4  (25)| 00:00:01 |
|   2 |   NESTED LOOPS                |           |     5 |  1140 |     3   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |           |     5 |  1140 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL         | ANIMAL    |     5 |    60 |     3   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN         | PK_OWNER  |     1 |       |     0   (0)| 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID| OWNER     |     1 |   216 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

SELECT STATEMENT = uskutočnil select dotaz
HASH GROUP BY = položky sa zoskupujú podľa hashovacieho kľúča.
NESTED LOOPS 2x = spojenie dvoch tabuliek (pre každú položku 1. tabuľky, všetky riadky druhej tabulky)
TABLE ACCESS FULL = prejdenie celou tabuľkou bez použitia indexov.
INDEX UNIQUE SCAN = prístup k tabuľkám cez B-strom. Získame jeden riadok podľa primárneho kľúča v tabuľke OWNER.
TABLE ACCESS BY INDEX ROWID = prístup k tabuľke cez index riadku

Potom sme vytvorili index

```
CREATE INDEX Animals ON animal (owner_id);
```

A spustili Explain plan znovu. Výsledkom bolo:

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |           |     5 |  1140 |     2  (50)| 00:00:01 |
|   1 |  HASH GROUP BY                |           |     5 |  1140 |     2  (50)| 00:00:01 |
|   2 |   NESTED LOOPS                |           |     5 |  1140 |     1   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |           |     5 |  1140 |     1   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | ANIMALS   |     5 |    60 |     1   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN         | PK_OWNER  |     1 |       |     0   (0)| 00:00:01 |
|   6 |    TABLE ACCESS BY INDEX ROWID| OWNER     |     1 |   216 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Použitím indexu sa podarilo znížiť celkovú cenu procesora zo 17 na 7.

3.4. Prístupové práva

Druhému členovi tímu boli udelené všetky práva pre prístup do tabuliek a spúšťať procedúry *speciesCount* a *titles.*

3.5. Materializovaný pohľad

Bol vytvorený materializovaný pohľad, *owned_pets,* ktorý zobrazuje všetky zvieratá, ktoré majiteľ vlastní. Pohľad sleduje tabuľky *owner* a *animal,* ktoré aktualizuje až po zavolaní príkazu *COMMIT.*

4. Záver

Skript sme vypracovali pomocou SQL Developer a JetBrains Datagrip. Na vypracovanie sme použili učebnú látku z predmetu IDS a rôzne internetové zdroje. Napr. www.tutorialspoint.com.