VYSOKÉ UČENÍ TECHNICKÉ V BRNE
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

DATABÁZOVÉ SYSTÉMY
2016/2017

FINÁLNY NÁVRH DATABÁZE
CECH ZLODEJOV

AUTORI PRÁCE
AUTHORS

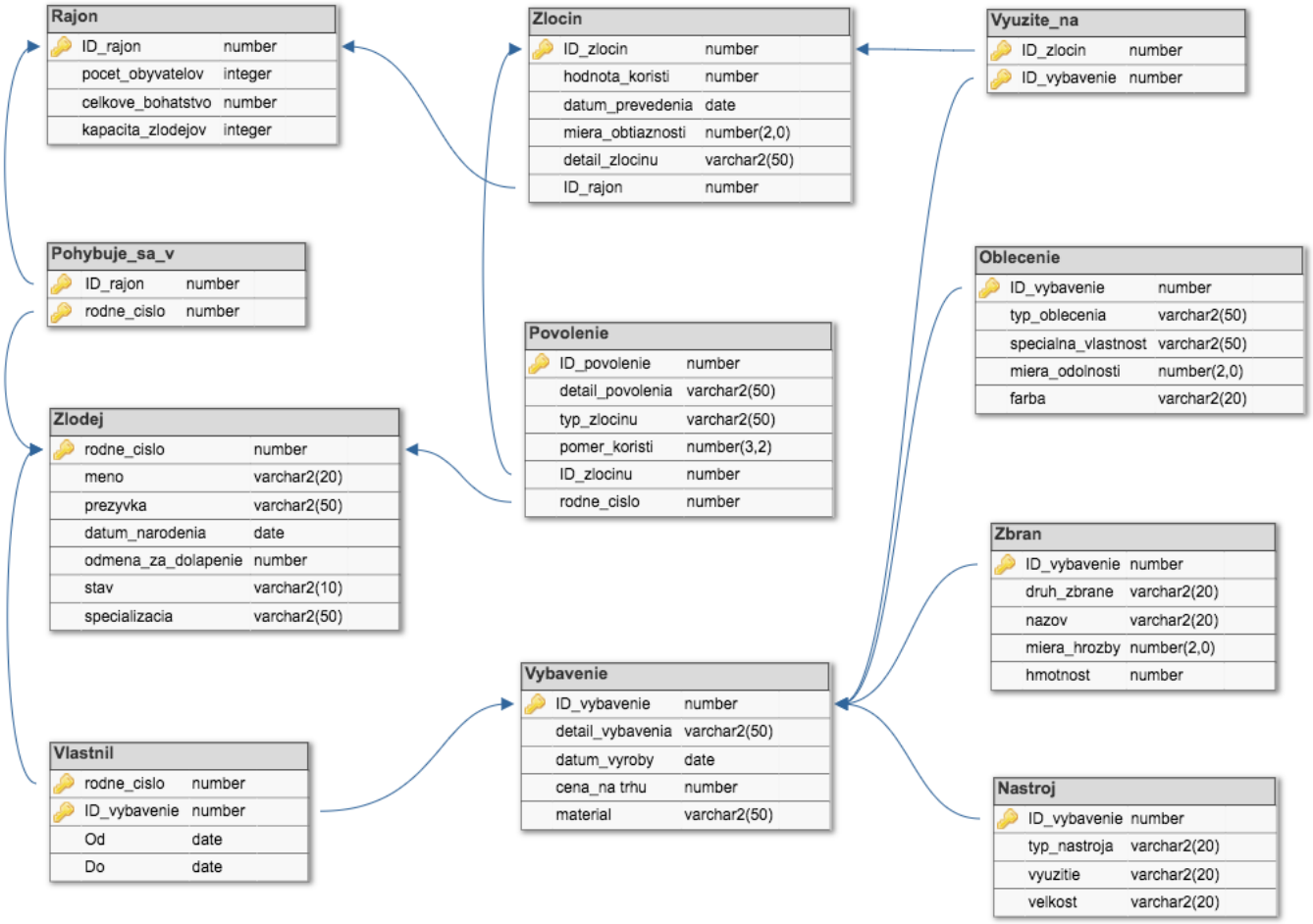
JAKUB HANDZUŠ (xhandz00)
MARTINA GRZYBOWSKÁ (xgrzyb00)

BRNO 29.4.2017

1. ZADANIE

Cech zlodejov chce zefektívniť svoju prácu a zadal výberové konanie na vytvorenie informačného systému pre evidenciu krádeží a lúpeží. Zlodeji zadávajú svoje rodné číslo, reálne meno (niektorí sú však bezmenní), prezývku (napríklad Vilda Dlouhoprsťák), dátum narodenia, stav, odmenu za dolapenie a naviac vlastnia celý rad vybavení, ktoré majú rôznu predajnú cenu na trhu a delia sa do troch rôznych kategórií - odevy, zbrane a nástroje. Pri oblečení je nutné evidovať jeho špeciálnu vlastnosť, pri nástroji jeho využitie a pri zbrani nás bude zaujímať miera hrozby, ktorú predstavuje. Vybavenie sa môže dediť a predávať ďalej (napríklad v prípade smrti), pričom evidujeme od kedy do kedy zlodej dané vybavenie vlastnil. Z dôvodu regulácie zločinu v meste sa vydávajú povolenia na uskutočnenie zločinu. Tieto povolenia sa uplatňujú na konkrétny typ zločinu, pričom sú jednorazové a neprenosné a taktiež určujú pomer zisku z koristi, ktorý zlodej po uskutočnení zločinu dostane. Je preto potrebné pri informáciach o zločine evidovať aj aké povolenie naň bolo využité. Vzhľadom na to, že zločin môže spáchať skupina zločincov, môže byť potrebné pri jednom zločine evidovať naraz viac povolení. Pri zločine nás ďalej bude zaujímať dátum prevedenia, hodnota lupu, vybavenie pri ňom využité a konkrétny rajón, v ktorom sa odohral. Každému zlodejovi sú pridelené rajóny, v ktorých bude lúpežne činný. O týchto rajónoch je potrebné uchovávať informácie týkajúce sa počtu obyvateľov, celkového bohatstva a kapacity zlodejov. Systém musí byť schopný pre užívateľa zlodeja vypísať zoznam zlodejov podľa odmeny za dolapenie, zoznam účastníkov konkrétneho zločinu a taktiež zoznam zločincov činných v konkrétnom rajóne.

2. Finálna schéma databázy



Obr.1 Finálny návrh databázy

3. Generalizácia/špecializácia

Generalizácia/špecializácia v našej databáze reprezentuje vzťah medzi entitnou množinou *Vybavenie* a entitnými množinami *Oblečenie*, *Zbrane* a *Nástroje*. V našej interpretácii zadania môže nastať prípad, kedy *Vybavenie* nespadá ani do jednej z týchto kategórií alebo naopak, spadá do viacerých súčasne. Z toho dôvodu sme ako vhodné prevedenie uvedeného vzťahu a súvisiacich entít do schémy relačnej databáze vybrali spôsob, pri ktorom vytvárame pre každú z týchto entít samostatnú tabuľku. Všetky tieto tabuľky zdieľajú spoločný primárny kľúč *ID_vybavenie*, pričom v tabuľkách *Oblečenie*, *Zbrane* a *Nástroje* sa zároveň jedná o cudzí kľúč odkazujúci sa do tabuľky *Vybavenie*.

4. Implementácia

Skript najprv vytvorí tabuľky zodpovedajúce dátovému modelu, zadefinuje určené integritné obmedzenia, triggery a následne tieto tabuľky naplní ukážkovými dátami. Na úplnom konci sú vykonávané operácie ako predvedenie niekoľkých základných dotazov SELECT, demonštrácia manipulácie s dátami pomocou uložených procedúr a optimalizácia pomocou vytvorenia indexu a jej zobrazenie cez EXPLAIN PLAN.

4.1 Triggery

Trigger na kontrolu správnosti zadaného rodného čísla sme implementovali podľa štandardných pravidiel. Ako prvá prebehne kontrola, či sa v zadanom čísle nachádzajú len numerické znaky. Nasleduje kontrola počtu znakov, ktorá tok rozvetvuje do troch možných smerov. Pokiaľ je počet znakov rovný 10, teda zlodej bol narodený po roku 1954, je testovaná deliteľnosť číslom 11, pokiaľ je rovný 9, je koncovka porovnávaná s reťazcom 000. V prípade, že sa počet znakov nerovná ani 10, ani 9, alebo došlo pri testoch vo vnútri podmienok k nejakej inej nezrovnalosti, skript zahlási chybu. Na úplnom konci skript testuje logickú správnosť samotného dátumu, ako správny počet mesiacov v roku a správny počet dní v mesiaci, vrátane kontroly prestupného roku.

Trigger na automatické generovanie hodnôt primárneho kľúča tabuľky *Povolenie* je realizovaný pomocou sekvencie, ktorá si pamätá číslo posledného pridaného kľúča. V prípade vygenerovania novej hodnoty sa toto číslo inkrementuje.

4.2 Procedúry

V skripte sa nachádzajú implementácie dvoch netriviálnych procedúr *pomerZlodejov()* a *rozdelenieLupuZlocinu()*, ktoré pre svoju činnosť využívajú kurzor pre prípadnú prácu s viacerými riadkami, premenné s dátovým typom odkazujúcim sa na riadok tabuľky, a taktiež ošetrujú možné vzniknuté výnimky.

Procedúra *pomerZlodejov()* po prijatí argumentu primárneho kľúča *ID_rajon* istého rajónu uloží do kurzoru ZLODEJI rodné čísla a mená všetkých zlodejov, ktorým tento rajón prislúcha. Ďalej pokračuje iterovaním cez všetky takéto riadky získanej tabuľky, pričom

vypisuje informácie o danom zlodejovi a zvyšuje hodnotu premennej reprezentujúcej počet zlodejov. Po dokončení cyklu si pomocou jednoduchého príkazu SELECT získa počet obyvateľov daného rajónu a vypočíta, koľko obyvateľov pripadá na jedného zlodeja. V prípade, že zadaný rajón neexistuje alebo v ňom nie sú činní žiadni zlodeji, a teda by sa delilo nulou, ošetrujúca výnimka nevyhodí chybu, iba informačné hlásenie.

Procedúra *rozdelenieLupuZlocinu()* po zadaní identifikátoru zločinu vypíše zoznam zločincov, ktorí sa na lupe podieľali a koľko si z neho odniesli. Suma sa počíta pre každého zlodeja zvlášť na základe celkovej hodnoty lupu a pomeru, ktorý zlodej dostane. Algoritmus procedúry je založený na kurzore, do ktorého sa nahrajú hodnoty z troch tabuliek a postupne sa iteruje cez každý riadok tabuľky. Po ukončení výpisu všetkých účastníkov zločinu sa taktiež vypíše celková ulúpená suma.

4.3 Materializovaný pohľad

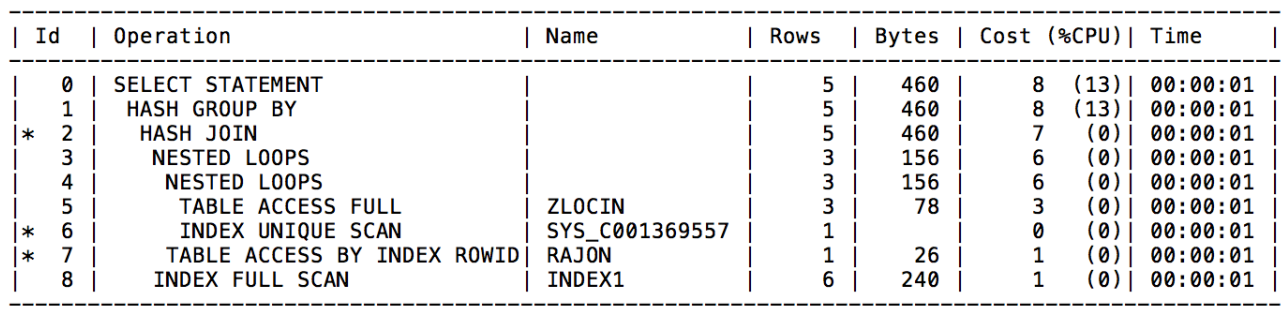
Pri vytváraní materializovaného pohľadu, ktorý využíva tabuľky definované jedným členom tímu, pričom patrí druhému členovi, sme danému pohľadu nastavili hneď na začiatku niekoľko vlastností. Najprv sme vytvorili maretrialozované logy, ktoré uchovávajú zmeny hlavnej tabuľky. Tieto logy nám umožňujú využiť vlastnosť REFRESH FAST ON COMMIT, ktorá zaisťuje aktualizáciu materilizovaných pohľadov po COMMIT-*e* práve na ich základe. Nasledujú vlastnosti CACHE, ktorá postupne optimalizuje čítanie z pohľadu, BUILD IMMEDIATE, ktorá naplní pohľad ihneď po jeho vytvorení a ENABLE QUERY REWRITE, pri ktorej bude materializovaný pohľad použitý pri optimalizácii dotazu, na ktorom bol založený. Po vytvorení samotného materializovaného pohľadu sme si najprv nechali vypísať jeho obsah, následne sme zmenili dáta v tabuľke, z ktorej bol tento pohľad vytvorený, a opäť sme vypísali jeho obsah. Tým, že jednotlivé výpisy sa navzájom líšili, sme si potvrdili funkčnosť nášho pohľadu.

4.4 EXPLAIN PLAN a vytvorenie indexu

Pomocou príkazu EXPLAIN PLAN sme demonštrovali optimalizáciu pri spracovaní dotazu zavedením indexu do tabuľky. V prvom prípade sme spustili dotaz nad tabuľkou, ktorá nevyužíva index, čoho výsledok je možné vidieť v prvom obrázku. V druhom prípade sme spúšťali rovnaký dotaz nad tabuľkou, ktorá už index využíva, vďaka čomu môžeme vidieť zníženie ceny operácii, avšak s navýšením procesorového času.

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	460	10 (10)	00:00:01
1	HASH GROUP BY		5	460	10 (10)	00:00:01
* 2	HASH JOIN		5	460	9 (0)	00:00:01
3	NESTED LOOPS		3	156	6 (0)	00:00:01
4	NESTED LOOPS		3	156	6 (0)	00:00:01
5	TABLE ACCESS FULL	ZLOCIN	3	78	3 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	SYS_C001369557	1		0 (0)	00:00:01
* 7	TABLE ACCESS BY INDEX ROWID	RAJON	1	26	1 (0)	00:00:01
8	TABLE ACCESS FULL	POVOLENIE	6	240	3 (0)	00:00:01

Obr.2 EXPLAIN PLAN bez použitia indexu



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5	460	8 (13)	00:00:01
1	HASH GROUP BY		5	460	8 (13)	00:00:01
* 2	HASH JOIN		5	460	7 (0)	00:00:01
3	NESTED LOOPS		3	156	6 (0)	00:00:01
4	NESTED LOOPS		3	156	6 (0)	00:00:01
5	TABLE ACCESS FULL	ZLOCIN	3	78	3 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	SYS_C001369557	1		0 (0)	00:00:01
* 7	TABLE ACCESS BY INDEX ROWID	RAJON	1	26	1 (0)	00:00:01
8	INDEX FULL SCAN	INDEX1	6	240	1 (0)	00:00:01

Obr.3 EXPLAIN PLAN s použitím indexu

Keďže pri tabuľkách, ktoré sú skromné na počet riadkov, nemusí byť pre databázu využitie indexov najoptimálnejším riešením kvôli väčšej náročnosti na procesorový čas, sme museli využitie indexu explicitne vyžiadať. Indexovanie sme zaviedli na stĺpec *ID_zločin* v tabuľke povolenie z dôvodu, že sa jedná o cudzí kľúč. Pre ešte väčšiu optimalizáciu sme zaviedli indexovanie na stĺpec *typ_zločinu*.

V následujúcich riadkoch popíšeme, ako systém spracuje a vykoná daný SELECT príkaz. Ako prvá operácia sa vykoná HASH GROUP BY, ktorá zoskupí riadky tabuľky na základe hashovacieho algoritmu. Pomocou operácie HASH JOIN sa spárujú záznamy spojovaných tabuliek cez hashovací kľúč spojenia. Následne sa dvakrát za sebou vykoná operácia NESTED LOOPS, pri ktorej sa vo vnorených cykloch každý riadok prvej tabuľky porovnáva s každým riadkom druhej tabuľky. Operácia TABLE ACCESS FULL prečíta celú tabuľku *Zločin*, všetky riadky a všetky stĺpce, bez ohľadu na indexy. INDEX UNIQUE SCAN hľadá rovnosť v stĺpci primárnych kľúčov, pričom pristupuje k tabuľkám cez B-strom. TABLE ACCESS BY INDEX ROWID získa konkrétny riadok z tabuľky *Rajón* pomocou indexu na primárny kľúč.

Posledná operácia sa líši v závislosti od toho či sa využil nami vytvorený index, alebo nie. V prvom prípade sa opäť využije TABLE ACCESS FULL, zatiaľ čo pri použití indexu sa prechádza na operáciu INDEX FULL SCAN, ktorá síce prečíta všetky hodnoty, ale len z indexovaných stĺpcov, miesto hodnôt zo všetkých stĺpcov celej tabuľky.

5. Pridelenie prístupových práv druhému členovi

Druhému členovi tímu boli udelené prístupové práva na všetky tabuľky pomocou príkazu GRANT ALL ON a takisto boli udelené práva na spúšťanie predpripravených procedúr pomocou príkazu GRANT EXECUTE ON.