Vysoké učení technické v Brne

Fakulta informačních technologií

Databázové systémy
2017/2018

Konceptuálny model

Truhlářství

Filip Pleško (xplesk02)
Tomáš Lapšanský (xlapsa00)

Brno, 29.04.2018

Obsah

Zadanie *3*

Schéma databáze *4*

Implementácia *5*

Trigger **5**

Check **5**

Procedúry **5**

Explain Plan a index **6**

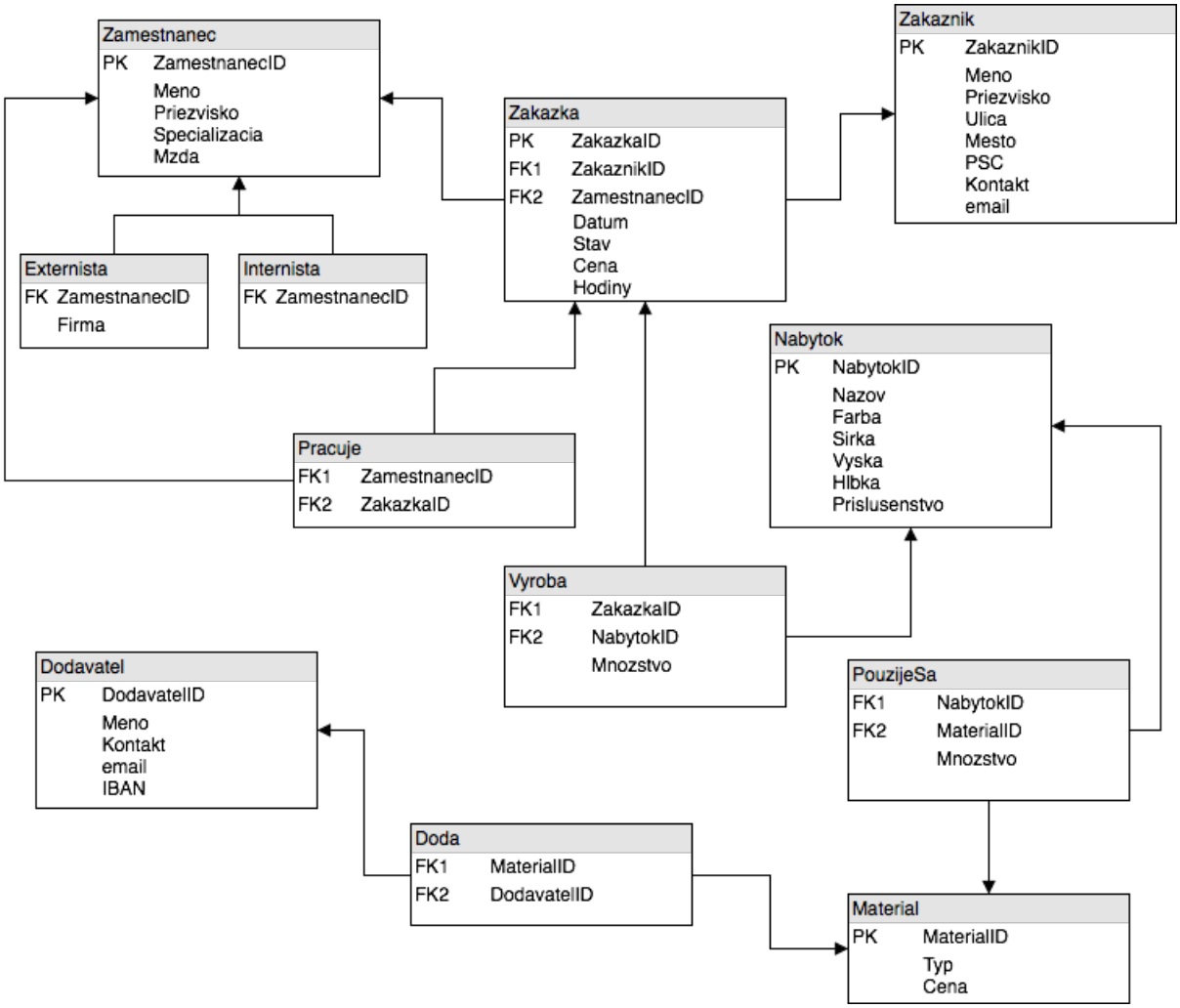
Pridelenie prístupových práv **6**

Materializovaný pohľad **6**

Zadanie

Navrhněte informační systém malého truhlářství, které přijímá zakázky od zákazníků. Každá zakázka má specifické požadavky. Každý kus nábytku má určen materiál, barvu, rozměry, příslušenství, atd. Systém uchovává informace o využitém materiálu na zakázce - materiál, množství, cena, atd. Firma odebírá materiál od více dodavatelů, přičemž každý může z dodavatelů může mít jiné ceny. Každá zakázka má zodpovědného zaměstnance, který řídí celou zakázku, přiděluje další zaměstnance na zakázku, určuje jaký materiál se použije apod. Na zakázce může pracovat více zaměstnanců; každý zaměstnanec má svoji specializaci. Firma může na zakázku najmout i externího zaměstnance, který má opět svoji specializaci a hodinovou mzdu. Jednotlivé položky na zakázce jsou fakturovány (použitý materiál, odpracované hodiny zaměstnanců, ...), z těchto položek je pak vyhotovena celková faktura zakázky.

Schéma databáze



Implementácia

Náš SQL skript mal podľa zadania vytvoriť na základe nami navrhnutého ER Diagramu základný model databáze, ktorú sme naplnili ukážkovými hodnotami. K nami vytvorenej databáze bolo potrebné dodať ukážkové SELECT skripty pre prácu s touto databázou. V ďalšej etape zadania bolo potrebné vytvoriť pokročilé SQL konštrukcie pre overenie správnosti nami vkladaných záznamov a pre automatické vytváranie určitých databázových položiek. Ďalej budú tieto skripty popísané nižšie.

Trigger

Medzi najzákladnejšie triggery v rámci našej implementácie určite patria triggery pre automatické generovanie primárnych kľúčov pre každý vložený prvok databázy. Pre vytvorenie takejto série triggerov sme využili nami definovanú sekvenciu nID, z ktorej sme čerpali pri každom vkladaní do ľubovoľnej tabuľky, a vyberali sme vždy takzvanú nextValue. Ďalší nami implementovaný trigger slúži pre kontrolu čísla IBAN v rámci tabuľky pre dodávateľov. V tomto triggery kontrolujeme rozsah IBAN-u, rozhodli sme sa implementovať rozsah 24-34 znakov pre kompatibilitu s medzinárodnými číslami IBAN, a ďalej kontrola znakov a čísel. Pre korektnú funkčnosť našej databáze sme sa rozhodli implementovať aj trigger pre tabuľku zákazka, kde je automaticky pridelený zodpovedný zamestnanec aj do sekcie zamestnancov pracujúcich na zákazke.

Check

Checky sme v našom SQL skripte využívali prevažne pre overenie správnosti atribútov v tabuľkách. Jedným z najlepších príkladov je overenie, či sa v mene a priezvisku zákazníka/zamestnanca nenachádza číslo, alebo obmedzenie veľkosti databáze pre ID na jeden milión.

Procedúry

Pre implementáciu procedúr sme museli podľa zadania použiť nasledovné: kurzor, ošetrenie výnimiek, a vytvoriť premennú ktorej dátový typ by sa odkazoval na typ riadku alebo stĺpca. Prvá procedúra ktorú sme sa rozhodli implementovať sčítava hodnoty ceny z tabuľky pre zákazky, no pre daný riadok musí platiť, že dátum jeho vytvorenia je v roku, ktorý sme procedúre zaslali ako parameter. Pre túto procedúru bolo potrebné využiť kurzor, kde sme vhodným selektorom vybrali riadny vyhovujúce našej podmienke, a neskôr cez tento kurzov iterovali. Ako druhú procedúru sme sa rozhodli zvoliť výpočet priemernej mzdy pre zamestnancov firmy. Znova sme využili kurzor do ktorého sme uložili tabuľku so zamestnancami, a potom sme následnou iteráciou zisťovali počet zamestnancov a súčet ich platov. Neskôr sme vypísali podiel týchto dvoch hodnôt, pri ktorom sme využili vstavanú funkciu round pre krajší výpis výsledku. V tomto prípade sme museli použiť výnimku pre delenie nulou, pretože táto podmienka by mohla nastať pri prechode prázdnou tabuľkou pre zamestnancov.

Explain Plan a index

Explain plan sme sa rozhodli demonštrovať na SELECT-e pre sčítanie počtu zakázok ktoré prislúchajú jednotlivým zákazníkom. Najprv sme sa rozhodli demonštrovať spustenie nášho skriptu bez použitia indexácie. Výsledok bol nasledovný:

ID	Operation	Name	Rows	Bytes	Cost(%CPU)	Time
0	SELECT STATEMENT		5	215	7 (15)	00:00:01
1	HASH GROUP BY		5	215	7 (15)	00:00:01
*2	HASH JOIN		5	215	6 (0)	00:00:01
*3	TABLE ACCESS FULL	ZAKAZKA	5	65	3 (0)	00:00:01
4	TABLE ACCESS FULL	ZAKAZNIK	6	180	3 (0)	00:00:01

Následne sme vytvorili index pre indexáciu ID cudzích kľúčov v Zakazka, výsledok bol nasledovný:

ID	Operation	Name	Rows	Bytes	Cost(%CPU)	Time
0	SELECT STATEMENT		5	215	4 (25)	00:00:01
1	HASH GROUP BY		5	215	4 (25)	00:00:01
2	NESTED LOOPS		5	215	3 (0)	00:00:01
3	TABLE ACCESS FULL	ZAKAZKA	6	180	3 (0)	00:00:01
*4	INDEX RANGE SCAN	PLAN_INDEX	1	13	0 (0)	00:00:01

Ako je možné vidieť, databáza v prvej fáze usúdila, že indexácia bude pre ňu zbytočne drahá operácia. My sme však indexáciu explicitne zadefinovali a ako je možné vidieť, aj napriek tomu že sa čiastočne zvýšilo percentuálne zaťaženie procesora, podarilo sa nám úspešne znížiť cenu jednotlivých operácií a v konečnom dôsledku nastalo urýchlenie.

Pridelenie prístupových práv

Pre pridelenie prístupových práv pre druhého člena tímu sme použili volanie GRANT ALL nad všetkými tabuľkami ktoré sme v našej databáze definovali. Týmto sme z ďalšieho užívateľa urobili v podstate administrátora tejto databáze, keďže bežný zamestnanec firmy nemá práva pristupovať k údajom svojich kolegov alebo nadriadených, a zákazník už vôbec nie.

Materializovaný pohľad

Pri implementácií takzvaného materializovaného pohľadu pre druhého užívateľa sme sa rozhodli využiť materializované logy nad tabuľkou ktorú budeme využívať. Rozhodli sme sa tak preto, aby sme mohli použiť tzv. FAST REFRESH ON COMMIT, a teda aby sa v materializovanom pohľade prejavili zmeny vždy po pridaní položky do databáze, a nemusel byť vždy vytváraný nový materializovaný pohľad. Táto operácia je prirodzene oveľa rýchlejšia ako nové vytváranie pohľadu. Pre demonštráciu sme sa rozhodli použiť jednoduchý SELECT. Ďalej sme pre demonštráciu vyššie spomínaného FROC pridali do tabuľky so Zákazkami novú

položku, zmeny potvrdili a spustili rovnaký SELECT skript znovu. Zmeny sa podľa očakávania okamžite prejavili. Okrem možnosti využitia FROC sme sa rozhodli použiť aj ďalšie prepínače pre zrýchlenie a zefektívnenie pohľadu.