VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Databázové systémy

2016/2017

Projekt 4-5. časť - Finální schéma databáze

Prodejna ze sýry

Martin Marušiak (xmarus07)

Juraj Ondrej Dúbrava (xdubra03)

Brno, 28. Apríl 2017

Zadanie:

Navrhněte IS prodejny se sýry, který by měl umožňovat evidovat zboží v prodejně a v jejím skladu. Dále by měl umožnit zaměstnanci objednávat sýry v prodejně a na skladu od různých dodavatelů. Jedna objednávka může obsahovat i více druhů zboží. Sýry se kategorizují podle více kritérií: země původu, živočicha (kravský, ovčí, kozí, jačí, ...), typu (ementálový, plísňový, uzený, ...).

U sýru je nutno evidovat procento tuku. Sýry jsou do prodejny dodávány v bochnících, kde pro každý bochník je třeba evidovat jeho počáteční hmotnost, aktuální hmotnost na prodejně, datum dodání, trvanlivost. Zaměstnanci si mohou vyhledávat sýry i podle země původu, u které jsou k dispozici relevantní informace o sýrech z dané země, či podle jeho druhu nebo živočicha.

Zaměstnanci prodejny mohou objednávat různé druhy sýrů u různých dodavatelů (ne všichni dodavatelé dodávají všechny druhy sýra), přičemž objednávka může obsahovat více druhů sýrů a u každého druhu je nutno specifikovat množství (hmotnost). Pro dodané sýry je nutno mít možnost dohledat, ke které objednávce patří a od kterého dodavatele jsou.

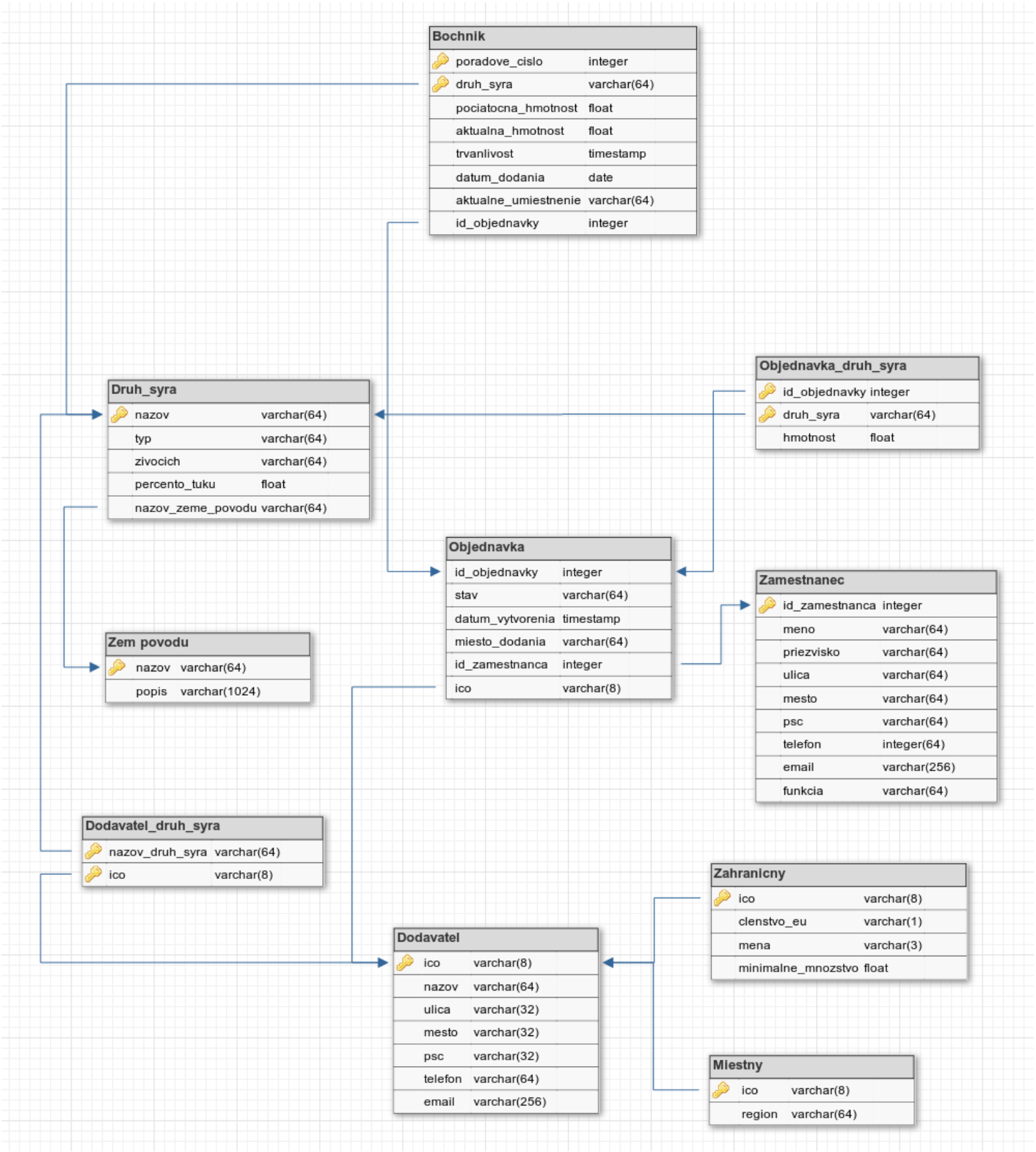
Dodatok zadania – generalizácia/špecializácia:

Dodávateľ môže byť lokálny alebo zahraničný. Pri zahraničnom dodávateľovi je potrebné uchovať informácie o členstve v EU, o peňažnej mene v ktorej prebehne platba objednávky a minimálnej hmotnosti pre objednávku. Pri lokálnom dodávateľovi si uchováme informáciu o regióne.

Generalizácia/špecializácia v schéme:

V našom prípade ide o generalizáciu/špecializáciu s úplnou príslušnosťou a disjunktnou špecializáciou. Pri prevode na schéma databáze sme sa rozhodli využiť 3 tabuľky, prvá reprezentuje spoločné vlastnosti – generalizáciu a ďalšie dve tabuľky predstavujú špecializácie na zahraničného a miestneho dodávateľa. Vzhľadom k tomu, že špecializácie sú disjunktné, tak máme možnosť realizácie pomocou dvoch tabuliek. Predpokladáme, že k spoločným údajom o dodávateľovi sa bude pristupovať častejšie ako k dátam, ktoré obsahujú špecializácie a preto sme sa rozhodli ponechať 3 tabuľky. Primárnym kľúčom všetkých troch tabuliek je IČO, ktoré je v prípade špecializácií zároveň cudzím kľúčom na generalizujúcu tabuľku dodávateľ. Schéma celej databáze je na nasledujúcej strane.

Schéma relačnej databázy



Obsah skriptu

SQL skript v sebe zahraná všetky predchádzajúce časti projektu, ktoré sa týkali predošlých úloh v jazyku SQL. Jedná sa o vytvorenie tabuliek, nastavenie integritných obmedzení, naplnenie tabuliek a dotazovanie nad tabuľkami. V tejto časti sme k skriptu pridali ďalšie databázové objekty, ako sú procedúry, triggre, sekvencie, index a materializovaný pohlaď. Taktiež sme si vyskúšali vysvetlenie dotazu a udelenie práv druhému členovi tímu.

Databázové triggre

V našom projekte používame 4 triggre, z toho 2 triggre sú určené pre automatické generovanie primárneho kľúča. Triggre na generovanie primárneho kľúča sa aktivujú pred operáciou insert, ak je vkladaná hodnota primárneho kľúča NULL. Uvedené triggre využívajú sekvenciu id_zamestnanca_seq a id_objednavky_seq. Ďalší trigger simuluje príkaz UPDATE ON CASCADE, ktorý v Oracle nie je podporovaný. Tento trigger má názov druh_syra_update a spolu s aktualizáciou názvu druhu syra v tabuľke DRUH_SYRA aktualizuje túto hodnotu aj v prípade tabuliek BOCHNIK a OBJEDNAVKA_DRUH_SYRA. Posledný trigger kontrola_ica kontroluje platnosť IČA pred jeho vkladaním do tabuľky a aktualizovaním. Trigger najskôr skontroluje formát IČA, teda skontroluje jeho dĺžku a povolené znaky – povolené sú len čísla. V ďalšej fáze sa vypočíta kontrolná číslica IČA na základe vzorca (11 – (8*n1 + 7*n2 + 6*n3 + 5*n4 + 4*n5 + 3*n6 + 2*n7) mod 11) mod 10, kde n1 je prvá číslica, n2 druhá, ... a n7 siedma. Ak je ôsma číslica IČA rovná vypočítanej kontrolnej číslici potom je IČO korektné.

Procedúry

Náš skript obsahuje dve procedúry a obe pracujú s bochníkmi. Obe procedúry spĺňajú požiadavky vyplývajúce za zadania. Teda obsahujú ošetrenie výnimiek, kurzory a taktiež využívajú %TYPE na typovanie premennej, ktorej typ sa odkazuje na typ stĺpca v tabuľke. Podobne sme využili aj %ROWTYPE na odkazovanie typu riadku tabuľky. Tento typ sme potom využili pri načítaní riadkov pomocou kurzoru do premennej.

Prvá procedúra pre prácu s bochníkmi vhodne_bochiky ma dva parametre, druh syra a hmotnosť. Procedúra má za úlohu vyhľadať bochníky určitého druhu syra, ktorých suma hmotností bude zhodná prípadne vyššia ako hmotnosť uvedená v parametri procedúry. Ak nemáme dostatočný počet bochníkov procedúra vypíše koľko kilogramov nám chýba, prípadne ak je suma hmotností bochníkov väčšia, koľko musíme z bochníka odrezať. Procedúra uprednostňuje bochníky, ktorým končí záruka skôr. Výstupom procedúry sú potrebné bochníky spolu s uvedením ich hmotnosti a trvanlivosti. Ak je uvedená hmotnosť v parametri záporná vyvolá sa výnimka, jej ošetrenie spočíva vo výpise chybovej hlášky. V prípade, že nie je k dispozícií žiadny bochník vyvolá sa vstavaná výnimka NO_DATA_FOUND a následne sa vypíše hláška informujúca užívateľa o tom, že z daného druhu syra neexistuje žiadny bochník.

Druhá procedúra skontroluj_dodane_bochniky kontroluje zhodu hmotností objednaného a doručeného syru. Táto procedúra je potrebná preto, lebo pri objednávke sa špecifikuje len druh syra a hmotnosť, to znamená, že neviem v koľkých a akých veľkých bochníkoch budú syry doručené, to závisí od dodávateľa. Preto je po prebraní objednávky a pridaní bochníkov do databázy, potrebné skontrolovať sumu

hmotností bochníkov a porovnať ju z objednanou hmotnosťou pre každý objednaný druh syra. V tejto procedúre využívame dva kurzory, prvý sprístupňuje objednané hmotnosti a v druhom sú sumy doručených bochníkov pre jednotlivé druhy syra. V procedúre postupne prechádzame oba kurzory a kontrolujeme zhodu v druhu a hmotnosti medzi doručenými bochníkmi a objednaným syrom.

Použitie indexu a explain plan

Vysvetlenie plánu prebiehalo na príkaze SELECT, ktorý má za úlohu vypísať sumu aktuálnych hmotností bochníkov daného typu a zoradiť ich zostupne podľa výslednej hmotnosti. Dotaz teda obsahuje spojenie dvoch tabuliek, agregačnú funkciu a zoradenie.

Najskôr sme si vypísali plán bez použitia indexu. Videli sme, že v prípade prístupu k tabuľke BOCHNIK sa využíva TABLE ACCES FULL, to znamená, že tabuľka bochník sa bude predchádzať celá bez použitia indexu. Vzhľadom k tomu, že bochníkov môže byť v rámci jedného druhu syra pomerne veľa, rozhodli sme sa pre túto tabuľku zaviesť index. Po zavedení indexu nad tabuľkou BOCHNIK, sme znova vypísali plán, tentokrát ale s hintom na použitie indexu. Ako vidíme v tabuľkách nižšie, naša nápoveda bola uplatnená a index bol naozaj použitý. Čo viedlo k zníženiu ceny v pláne.

Vysvetlenie dotazu bez použitia indexu

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10	1150	5 (40)	00:00:01
1	SORT ORDER BY		10	1150	5 (40)	00:00:01
2	HASH GROUP BY		10	1150	5 (40)	00:00:01
3	NESTED LOOPS		10	1150	3 (0)	00:00:01
4	NESTED LOOPS		10	1150	3 (0)	00:00:01
5	TABLE ACCESS FULL	BOCHNIK	10	470	3 (0)	00:00:01
6	INDEX UNIQUE SCAN	PK_NAZOV_DRUH_SYRA	1		0 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	DRUH_SYRA	1	68	0 (0)	00:00:01

Vysvetlenie dotazu s hintom na použitie indexu

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10	1150	4 (50)	00:00:01
1	SORT ORDER BY		10	1150	4 (50)	00:00:01
2	HASH GROUP BY		10	1150	4 (50)	00:00:01
3	NESTED LOOPS		10	1150	2 (0)	00:00:01
4	NESTED LOOPS		10	1150	2 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID BATCHED	BOCHNIK	10	470	2 (0)	00:00:01
6	INDEX FULL SCAN	BOCHNIK_DRUH_SYRA_IX	10		1 (0)	00:00:01
7	INDEX UNIQUE SCAN	PK_NAZOV_DRUH_SYRA	1		0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	DRUH_SYRA	1	68	0 (0)	00:00:01

V nasledujúcom texte popisujeme jednotlivé riadky z výpisu plánu. Z prvého riadku explain plan (id 0), môžeme zistiť, že sa jedná o príkaz SELECT. Ďalší riadok SORT ORDER BY nás informuje o tom, že výsledok bude zoradený z dôvodu použitia klauzule ORDER BY v dotaze. Ďalej tu je HASH GROUP BY, ktorá nás informuje o metóde použitej pri agregácií, konkrétne sa jedná použitie hašovania. Potom nasleduje dvakrát NESTED LOOPS, tieto operácie súvisia so spájaním tabuliek a indikujú, že pri spájaní

sa použijú zanorené cykly, táto metóda je užitočná, keď je počet položiek v tabuľkách malý. Jedná sa o porovnanie každého riadku prvej tabuľky s každým riadkom tabuľky druhej.

V ďalšom kroku sa plán s použitím indexu a plán bez indexu líši. V prípade plánu bez indexu, je typ prístupu k tabuľke bochník TABLE ACCESS FULL, čo znamená, že sa bude prechádzať celá tabuľka bez použitia indexu, riadok po riadku. Naproti tomu v prípade použitia indexu sa tento prístup zmení a pribudnú operácie TABLE ACCESS BY INDEX ROWID a INDEX FULL SCAN. TABLE ACCESS BY INDEX ROWID BATCHED značí, že sa jedná o prístup k riadku s využitím načítaného indexu. INDEX FULL SCAN je metóda prístupu cez B-strom (B-Tree Index Acces) a zabezpečuje prístup k dátam s pomocou indexu, prípadne dokáže získať potrebné informácie priamo z indexu. V stĺpci name môžeme vidieť, že sa použil náš index nad tabuľkou BOCHNIK.

V nasledujúcich krokoch sú plány rovnaké. Použije sa znova metóda prístupu cez B-strom, tentokrát ale namiesto FULL SCAN bude použitá metóda UNIQUE SCAN pre získanie hodnoty jedného riadku. Táto metóda je vhodná v prípade porovnávania na rovnosť u stĺpcovou s unikátnymi hodnotami. V našom prípade je toto porovnanie následkom spájania tabuliek DRUH_SYRA a BOCHNÍK. V poslednom kroku je opäť použitá metóda TABLE ACCESS BY INDEX ROWID.

Prístupové práva

Druhému členovi tímu boli priradené všetky práva a to na všetky tabuľky, ktoré sme vytvorili. Práva boli priradené pomocou príkazu GRANT ALL PRIVILEGES nad danými tabuľkami. Druhý člen tímu, ďalej získal privilégium na spustenie procedúr – EXECUTE, jedná sa o procedúry vhodne_bochniky a skontroluj_dodane_bochniky. Na získanie hodnoty sekvencií id_zamestnanca_seq a id_objednavky_seq bolo druhému členovi priradené privilégium SELECT nad danými sekvenciami. Ešte pred zavedením oprávnení sme overili, že druhý člen nemá prístup k tabuľkám prvého člena, následne sme práva udelili a otestovali prístup k tabuľkám, procedúram a sekvenciám. Druhý člen tímu pristúpil k tabuľkám niekoľkými spôsobmi. Najskôr sme zmenili hodnotu aktuálnej schémy na schému prvého člena tímu pomocou ALTER SESSION SET CURRENT_SCHEMA. Potom sme vyskúšali prístup na základe uvedenia názvu schémy prvého člena tímu pred názvy tabuliek spolu s vytvorením synonyma.

Materializovaný pohľad

Obsah nášho materializovaného pohľadu je založený na jednoduchom dotaze, ktorý spája tabuľky DRUH_SYRA a BOCHNÍK. Uchovávame v ňom informácie o bochníku, jeho hmotnosti, trvanlivosti, poradovom čísle, type a druhu. Pri tvorbe pohľadu sme využili nasledujúce klauzule: NOLOGGIN, CACHE, BUILD IMMEDIATE, REFRESH FAST ON COMMIT a ENABLE QUERY REWRITE.

Klauzula NOLOGGING zapríčiní, že sa nebudú zaznamenávať operácie z pohľadom do redu-logu, čím sa samozrejme zníži réžia pri práci s pohľadom. Ďalej sme použili CACHE, ktorý slúži na optimalizáciu prístupu k používaním blokom. BUILD IMMEDIATE zabezpečuje okamžité naplnenie pohľadu hneď po jeho vytvorení. Pre rýchlu obnovu (refresh fast) po potvrdený transakcie (on commit) sme použili klauzulu REFRESH FAST ON COMMIT. Toto nastavenie spôsobí, že sa pri aktualizácií dát v pohľade znova nevyhodnotí daný dotaz, ale využijú sa redu-logy master tabuliek. Z toho vyplýva potreba vytvorenia týchto logov pre master tabuľky, teda tabuľky, ktoré sme použili v dotaze pohľadu. Konkrétne sa jedná o vytvorenie logov pre tabuľku DRUH_SYRA a pre tabuľku BOCHNÍK. Tieto logy sme vytvorili pred vytvorením materializovaného pohľadu. Ako posledné sme použili ENABLE QUERY REWRITE, ktorý

umožní využitie pohľadu optimalizátorom. Potom boli druhému členovi tímu priradené práva pre prácu s týmto pohľadom. Následne sme nad materializovaním uskutočnili pár jednoduchých dotazov. Skúsili sme aktualizovať hodnotu v master tabuľke a potom sme vypísali hodnoty z pohľadu pomocou dotazu, hodnota bola v pohľade nezmenná. Následne sme vykonali príkaz potvrdenia transakcie – COMMIT a znova zadali dotaz nad pohľadom. Tentokrát už obsahoval aktualizovanú hodnotu čím sme demonštrovali nastavenie ON COMMIT.

Záver

Na tvorbu SQL skriptu sme využívali nástroj SQL Developer. Za účelom spustenia a testovania skriptu sme využívali školský server Oracle 12c. Skripty sme vypracovali na základe obsahu prednášok, democvičení a ukážkových skriptov dostupných na stránke IDS. Ako doplnkový zdroj sme využívali rôzne články, tutoriáli a oficiálnu Oracle dokumentáciu.