

Databázové systémy 2018/2019

Dokumentácia projektu: zadanie Reštaurácia

Zadanie

Restaurace poskytuje běžné stravovací služby veřejnosti. Mimo to umožňuje pořádání akcí v saloncích na základě rezervace. V restauraci je možné rezervovat i jednotlivá místa. Navrhněte informační systém, který podpoří běžné činnosti podniku.

1. Tvorba výslednej schémy

Na základe informácií získaných na prednáškach sme vytvorili ER model databázy. Model databázy z prvej fázy projektu sme previedli na schéma databázy v jazyku PL/SQL. Celý projekt bol tvorený pomocou programu SQL Developer od firmy Oracle. Na vytvorenie schémy sme využili príkaz *CREATE TABLE*. Následne sme pridali atribúty tabuliek a na základe zadania projektu sme taktiež aplikovali dve obmedzenia pomocou príkazu *CONSTRAINT*. Dáta sme vkladali pomocou príkazu *INSERT*.

2. Triggery

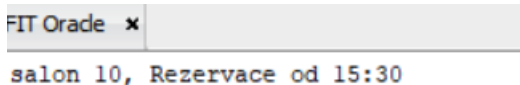
V našej schéme sme vytvorili dva triggery, *osoba_trigger* a *rezervace_mista_trigger*. *osoba_trigger* automaticky vytvára ID osoby pri jej vkladaní do systému. *rezervace_mista_trigger* tvorí rezerváciu miesta, za predpokladu že tento termín je dostupný. Nakoľko nevieme určiť presnú dĺžku pobytu zákazníka, explicitne sme ju nastavili na 2 hodiny. Týmto pádom sa nedá rezervovať v období 1 hodiny a 59 minút pred alebo po nájdení existujúcej rezervácie.

3. Procedúry

Taktiež sme vytvorili dve procedúry, *Obsazenost* a *Rezervace*.

Prvá procedúra kontroluje obsadenosť a vypisuje miesta, ktoré sú v zadanom dátume a časovom rozmedzí obsadené. Táto procedúra by sa dala použiť pre zobrazenie voľných a obsadených miest, napríklad pri tvorení online rezervácie. Pre správne prevedenie tejto procedúry je potrebné aby bol počiatočný čas skorej ako ten koncový. Pokiaľ užívateľ túto podmienku nespĺní, program vypíše chybu. Ako už bolo spomínané, aj v tomto prípade používame „bežnú dobu návštevy“, teda 2 hodiny.

Druhá procedúra môže slúžiť ako tvorba ceduliek, ktoré budú uložené na danom mieste. Táto procedúra automaticky vytvorí text ceduliek pre aktuálny deň. V reálnom systéme sme očakávame, že sa rezervácie na daný deň uzatvárajú aspoň jeden deň vopred. Do kurzoru uložíme všetky rezervácie s dnešným dátumom. Potom pomocou cyklu prejdeme všetky dané rezervácie a vytvorí k nim odpovedajúce cedulky.



```
FIT Orade x
salon 10, Rezervace od 15:30
```

Výstup procedúry *Rezervace*

ID	TYP
8	salon
3	miesto
4	miesto

Výstup procedúry Obsazenost

4. Explain plan

Explain plan zobrazuje spracovávanie daného príkazu. Vybrali sme *SELECT* príkaz, ktorý zobrazuje počet objednávok jednotlivých zákazníkov. Ten spája tri tabuľky, a to *R_Osoba*, *R_Zakaznik* a *R_Objednavka*. Taktiež obsahuje klauzulu *GROUP BY* agregáciu funkciu *COUNT*. Túto klauzulu využívame dva krát – pre zobrazenie účinnosti optimalizácie pomocou *INDEXu*.

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10	800	7 (15)	00:00:01
1	HASH GROUP BY		10	800	7 (15)	00:00:01
* 2	HASH JOIN		10	800	6 (0)	00:00:01
3	TABLE ACCESS FULL	R_OSoba	6	402	3 (0)	00:00:01
* 4	TABLE ACCESS FULL	R_OBJEDNAVKA	10	130	3 (0)	00:00:01

Výstup klauzule *EXPLAIN PLAN* pred optimalizáciou

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		10	800	1 (0)	00:00:01
1	SORT GROUP BY NOSORT		10	800	1 (0)	00:00:01
2	NESTED LOOPS		10	800	1 (0)	00:00:01
3	INDEX FULL SCAN	OSoba_INDEX	6	402	1 (0)	00:00:01
* 4	INDEX RANGE SCAN	OBJEDNAVKA_INDEX	2	26	0 (0)	00:00:01

Výstup klauzule *EXPLAIN PLAN* po optimalizácii

5. Index

Pre zlepšenie výkonu databáze sme vytvorili dva indexy na stĺpce, pri ktorých očakávame najvyšší počet prístupov. Prvý *INDEX*, *osoba_index*, indexuje stĺpce *ID*, *MENO* a *PRIEZVISKO* z tabuľky *R_Osoba*. Druhý *INDEX*, *objednavka_index*, indexuje stĺpec *VYTVORIL_ZAK* z tabuľky *R_Objednavka*. Indexovanie znižuje cenu prevedenia daného príkazu (využitie procesora).

6. Garantovanie práv druhému členovi tímu

Na prístup druhého člena do tabuľky R_Rezervuje bol použitý príkaz *GRANT INSERT ON*.

7. Materializovaný pohľad

Vytvorili sme materializovaný pohľad na tabuľku R_Rezervuje vrátane údajov o osobách z tabuľky R_Osoba pre druhého člena tímu, teda užívateľa XLINKA01. Pomocou príkazu *CACHE* sa optimalizuje doba prístupu k dátam. *BUILD IMMEDIATE* napĺňa pohľad dátami ihneď po vytvorení. *REFRESH ON COMMIT* aktualizuje dáta pohľadu pri každom uložení dát do databázi a následnom uložení pomocou príkazu *COMMIT*.

Použitie materializovaného pohľadu demonštrujeme na príkaze *SELECT*, ktorý vypíše aktuálny obsah pohľadu. Potom užívateľ XLINKA01 vloží do systému novú rezerváciu a následne ďalším príkazom *SELECT* ukáže, že sa dáta v tabuľke nezmenili. Zmena nastane až následujúcim príkazom *COMMIT*. Posledný príkaz *SELECT* už ukáže aktualizované dáta pohľadu.

```
SELECT * FROM XJURIG00.REZERVACE_MV; -- Vypise aktualni pohled

INSERT INTO XJURIG00.R_Rezervuje(DATUM, CAS, POCET_OSOB, ZAKAZNIK, MIESTO) VALUES (TO_DATE('2019-04-29','YYYY-MM-DD'),TO_DATE('13:37', 'HH24:MI'), '1', '4', '4');
-- Vlozi rezervaci

SELECT * FROM XJURIG00.REZERVACE_MV; -- Vypise porad puvodni pohled

COMMIT; -- Ulozi zmeny v databazi

SELECT * FROM XJURIG00.REZERVACE_MV; -- Vypise upraveny pohled obsahujici novou polozku tabulky
```

Demonštrácia použitia materializovaného pohľadu užívateľom XLINKA01