VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Databázové systémy
2016/2017
Konceptuální model
Restaurace

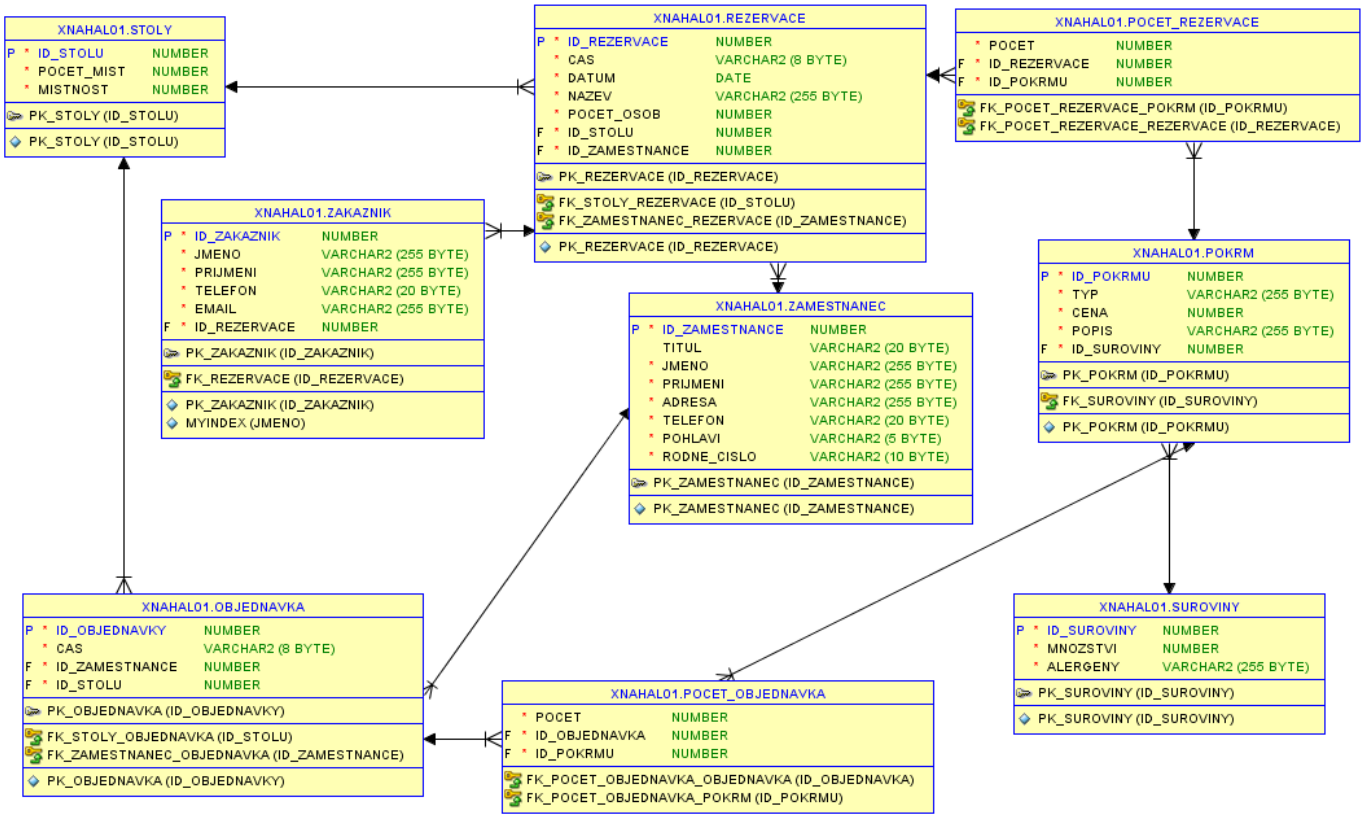
Roman Nahálka (xnahal01)
Martin Mihál(xmihal07)

1. května 2017

1 Zadání

Vytvořte IS pro restaurační zařízení, který napomůže k zjednodušení a zpřehlednění jeho provozu. Restaurace je členěna do více místností a má přední a zadní zahrádku a poskytuje běžné stravovací služby veřejnosti. Od informačního systému se požaduje, aby, krom jiného, umožnil správu rezervací a objednávek. Rezervovat je možné jeden nebo více stolů v místnostech či na zahrádkách, anebo celé místnosti, případně i celou restauraci pro různé společenské akce. Součástí rezervace také může být objednávka nápojů a jídel. Systém musí umožňovat zaměstnancům restaurace vkládat objednávky spolu s informacemi, který zaměstnanec danou objednávku vložil a pro koho je určena. Když se zákazníci rozhodnou zaplatit, musí jim systém vystavit účtenku. Po zaplacení pak příslušný zaměstnanec vloží záznam o platbě do systému. Systém by měl také poskytovat podrobný přehled tržeb za vybrané období. Přístup k této funkci bude mít pouze majitel. V neposlední řadě musí systém evidovat veškeré prodávané jídlo a pití (včetně složení), přičemž majitel a odpovědný vedoucí mají možnost měnit ceny jídla a pití nebo přidávat a odebírat položky.

2 Schéma databáze



3 Implementace

Byl navrhnut jednoduchý informační systém pro správu restaurací. Systém umožňuje evidenci objednávek, zaměstnanců, zákazníků, stolů, pokrmů a surovin. Systém také umožňuje správu rezervací. K těmto položkám mají právo zasahovat pouze zaměstnanci restaurace.

3.1 Triggery

Skript obsahuje celkem tři triggery. V prvních dvou případech se jedná o automatickou inkrementaci primárního klíče, konkrétně v tabulkách Rezervace a Objednávka. Číslování těchto triggerů začíná na čísle 1. Třetím a posledním triggerem je kontrola rodného čísla v tabulce Zaměstnanec. Tento trigger při vložení nového zaměstnance do tabulky zkontroluje, že zadané rodné číslo zaměstnance je platné. Kontroluje tedy, že má platný rozsah 9–10 čísel a zadané hodnoty pro rok, den a měsíc narození jsou v pořádku.

3.2 Procedury

Náš skript obsahuje celkem dvě procedury. První procedura je na výpis počet obsloužených zákazníku pro konkrétního zaměstnance. Druhá procedura vypíše počet rezervací, uskutečněných konkrétním zákazníkem.

- `pocet_obslouzenych_zakaznikov` – Procedura, která má jako vstupní parametr id zaměstnance výstup je počet obsloužených lidí tímto zaměstnancem. V případě neexistujícího zaměstnance je ošetřena výjimka.
- `pocet_rezervaci` – Procedura, která má jako vstupní parametry jméno a příjmení zákazníka a výstup je počet rezervací provedených tímto zákazníkem. V případě neexistujícího zákazníka je ošetřena výjimka.

3.3 Ukázkové výběry dat z databáze

Vytvořili jsme následující ukázky výběru dat z databáze pro demonstraci funkčnosti databáze.

- Výpis objednávky, zaměstnance, který objednávku vytvořil a stolu.
- Výpis typu pokrmu a počet pokrmů stejného typu.
- Výpis zaměstnanců a počet objednávek, které mají přidělené.
- Výpis zaměstnanců, kteří nemají přidělenou objednávku.
- Výpis rezervací, kde je rezervovaný stůl s dvěma či více stoly.
- Výpis jídla a jeho surovin.
- Výpis zákazníků a jejich rezervací.

3.4 Explain plan a vytvoření indexu

Podle zadání jsme použili EXPLAIN PLAN pro výpis plánu provedení databázového dotazu. EXPLAIN PLAN jsme použili nad jednoduchým SELECTEM, který spojuje dvě tabulky, obsahuje agregační funkci a používá klauzuli GROUP BY. Nejprve jsme spustili EPLAIN PLAN bez použití indexu a poté s indexem.

Id	Operation	Name	Rows	Bytes	Cost(% CPU)	Time
0	SELECT STATEMENT		3	504	4 (25)	00:00:01
1	HASH GROUP BY		3	504	4 (25)	00:00:01
2	NESTED LOOPS		3	504	3 (0)	00:00:01
3	NESTED LOOPS		3	504	3 (0)	00:00:01
4	TABLE ACCESS FULL	ZAKAZNIK	3	426	3 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PK_REZERVACE	1		0 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	REZERVACE	1	26	0 (0)	00:00:01

Tabulka 1: EXPLAIN PLAN bez použití indexu

Id	Operation	Name	Rows	Bytes	Cost(% CPU)	Time
0	SELECT STATEMENT		3	504	3 (34)	00:00:01
1	HASH GROUP BY		3	504	3 (34)	00:00:01
2	NESTED LOOPS		3	504	2 (0)	00:00:01
3	NESTED LOOPS		3	504	2 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID BATCHED	ZAKAZNIK	3	426	2 (0)	00:00:01
5	INDEX FULL SCAN	MYINDEX	3		1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	PK_REZERVACE	1		0 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	REZERVACE	1	26	0 (0)	00:00:01

Tabulka 2: EXPLAIN PLAN bez použití indexu

Nejdříve jsme EXPLAIN PLAN zavolali bez použití námi vytvořeného indexu a poté jsme EXPLAIN PLAN zavolali znovu s použitím indexu. Jak můžeme vidět ve výše uvedených tabulkách, tak se snížila položka `Cost`, tedy počet zdrojů potřebných pro vykonání plánu, čím nižší je tato položka, tím efektivnější plán je. Naopak se nám zvýšilo vytížení CPU. Z tabulky lze dále vyčíst, že při použití indexu se provede jedna operace navíc a 4. operace je nahrazena jinou. Rozebereme si jednotlivé operace v EXPLAIN PLAN:

- `SELECT STATEMENT` – Znamená, že byl uskutečněný dotaz SELECT.
- `HASH GROUP BY` – Znamená, že se dotaz seskupí podle hashovacího klíče.
- `NESTED LOOPS` – Znamená, že spojované tabulky prohledává ve vnořených cyklech, tedy že se každý řádek první tabulky porovnává se všemi řádky druhé tabulky, atd.
- `TABLE ACCESS FULL` – Pouze u plánu bez indexu. Znamená, že se prochází celá tabulka od začátku bez použití indexů.
- `TABLE ACCESS BY INDEX ROWID BATCHED` – Pouze u plánu s indexem. Znamená, že se přistupuje do tabulky přes konkrétní řádek pomocí námi vytvořeného indexu.
- `INDEX FULL SCAN` – Pouze u plánu s indexem. Znamená, že provádí výpis hodnot z indexovaných sloupců.

- `INDEX UNIQUE SCAN` – Znamená, že se přistupuje k tabulkám přes B-strom, kdy nám vypadne jedinečný řádek podle primárního klíče, v tomto případě podle primárního klíče `PK_REZERVACE`.
- `TABLE ACCESS BY INDEX ROWID` – Znamená, že se přistupuje do tabulky přes index.

3.5 Přístupová práva

V rámci projektu byli vytvořeny přístupová práva pro druhého člena týmu. Druhému členu týmu byl udělen přístup ke všem tabulkám a procedurám databáze.

3.6 Materializovaný pohled

Byl vytvořen materializovaný pohled patřící druhému členu týmu, který musel mít nastavené přístupové práva. Optimalizace pohledu proběhla pomocí logů. Díky tomu se nemusí celá tabulka obnovovat, ale zachytávají se jenom změny a ty se aktualizují. Následně byl vytvořen pohled. Použili jsme následující způsoby optimalizace:

- `CACHE` – Využije se cache paměť pro rychlejší načtení dat.
- `BUILD IMMEDIATE` – Hned po vytvoření se pohled naplní daty.
- `REFRESH FAST ON COMMIT` – Rychlá aplikace změn.
- `ENABLE QUERY REWRITE` – Použitelnost pohledu pro optimalizátor.