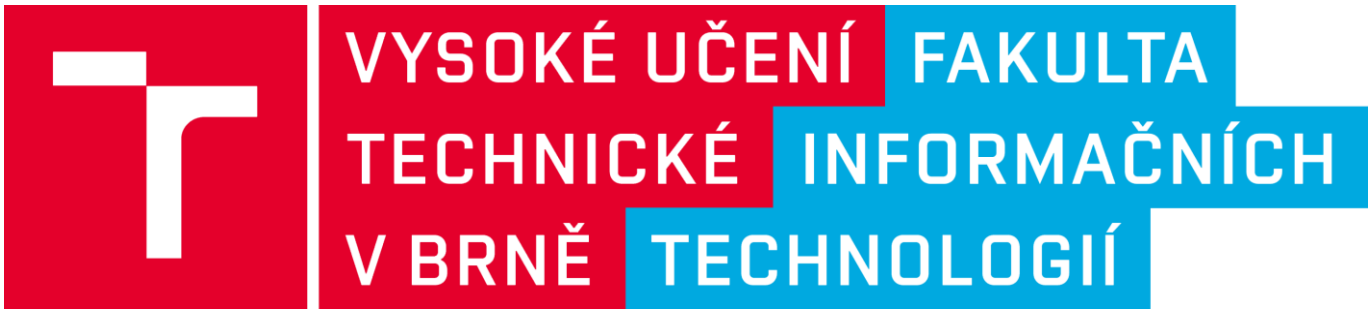
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Dokumentace k projektu do předmětu IDS

SQL skript

Tým xillic00_xfryct00

Zadání z předmětu IUS – Hudební festivaly a koncerty

Členové:

Illichmann David **xillic00**

Fryč Tomáš **xfryct00**

Úvod

V následujících řádcích popisujeme finální řešení schématu databáze projektu do předmětu IDS. Cílem našeho projektu bylo vytvoření databáze pro uchovávání dat hudebních festivalů a koncertů, ceny a informace o vstupenkách na tyto akce, informace interpretů a údaje uživatelů.

Základní objekty

V první části skriptu je vytvořeno několik tabulek pro jednotlivé části databáze a nastavuje primární a cizí klíče. Poté vkládá ukázková data do těchto tabulek. Následuje ukázková práce s daty. Toto vše byly předchozí části projektu.

Triggery

Náš první trigger ověřuje korektní formát zadaného čísla vstupenky *cislo_vstupenky* v tabulce *vstupenka.* Tento trigger nejprve kontroluje, zda je na prvních 9 místech čísla 9 číslic, poté dalších sedm pozic musí obsahovat písmena a po nich osm číslic. Náš další trigger je na autoinkrementaci primárního klíče *id_performer* každého interpreta přidaného do tabulky *interpret.* Tento trigger je realizovaný pomocí sekvence, k zapamatování si posledního čísla.

Procedury

Vytvořili jsme proceduru *pocetLike*, která vezme zadané id interpreta, najde název tohoto interpreta v jeho tabulce a poté spočítá počet výskytu tohoto id v tabulce *mit_v_oblibe*. Výsledné číslo je uloženo v proměnné *counter*, která se inkrementuje při každém nalezení shody. Procedura nakonec vypíše jméno interpreta s výsledným počtem „fanoušků“. V případě, že při své práci procedura narazí na výjimku *NO_DATA_FOUND* vypíše chybovou hlášku a znamená to, že zadanému id neexistuje žádný interpret.

Druhá procedura *pocetKoncertu* pracuje znovu s id interpreta a dvěma daty. Tato procedura vyhledá jméno interpreta podle id v tabulce *interpret*, poté prochází spojení tabulek *interpret* a *poradi_kapel* a pokud nalezne záznam, ve kterém zadaný interpret vystupuje v rozmezí dvou dat, inkrementuje počítadlo. Na konci procedury se výsledný počet koncertů v daném období vypíše. Pokud zadaný interpret neexistuje, následuje výjimka a její chybová hláška.

Explain plan a vytvoření indexu

Náš explain plan používá jednoduchý select dotaz *telefon* pro vypsání všech telefonních čísel zákazníků z tabulky zakaznik, kteří mají v oblibě nějakého interpreta. Ten nejprve spouštíme bez použití indexu. Poté vytváříme index a spouštíme explain plan znovu.

Z výsledku je patrné, že se snížil cost (přístupy na disk), ale na druhu stranu se zvýšilo využití CPU.

Před použitím indexu:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		15	390	10 (10)	00:00:01
1	HASH GROUP BY		15	390	10 (10)	00:00:01
2	MERGE JOIN CARTESIAN		15	390	9 (0)	00:00:01
3	TABLE ACCESS FULL	ZAKAZNIK	3	39	3 (0)	00:00:01
4	BUFFER SORT		5	65	7 (15)	00:00:01
5	TABLE ACCESS FULL	MIT_V_OBLIBE	5	65	2 (0)	00:00:01

Po použití indexu:

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		15	390	7 (15)	00:00:01
1	HASH GROUP BY		15	390	7 (15)	00:00:01
2	MERGE JOIN CARTESIAN		15	390	6 (0)	00:00:01
3	TABLE ACCESS FULL	ZAKAZNIK	3	39	3 (0)	00:00:01
4	BUFFER SORT		5	65	4 (25)	00:00:01
5	INDEX FAST FULL SCAN	INDEXEXPLAINTMP	5	65	1 (0)	00:00:01

Přístupové práva

Přístupové práva k jednotlivým tabulkám databáze a procedurám jsou uděleny uživateli xfryct00. Uživatel má neomezený přístup.

Materializovaný pohled

Materializovaný pohled patřící druhému členovi týmu (xfryct00), který používá tabulky definované uživatelem xillic00 nejprve vytváří materializované logy. Zde se uchovávají změny tabulky. Tyto logy jsou vytvořeny, aby se mohl při změnách tabulky využívat fast refresh on commit oproti complete refresh (musel by se znovu spouštět celý dotaz pohledu, což by trvalo déle). Po vytvoření logů je vytvořen samotný pohled (view), kterému nastavujeme kromě refresh fast on commit i tyto další vlastnosti: cache (optimalizuje to, co se načítalo z pohledu), build immediate (vytvoří se pohled a ihned naplní), enable query rewrite (materializovaný pohled bude používaný optimalizátorem).

Následuje předvedení funkčnosti materializovaného pohledu na příkladu. Nejprve se vypíše, co všechno materializovaný pohled obsahuje, poté jsou přidány nová data do tabulky (ze které byl pohled vytvořen), následuje příkaz commit a nakonec se znovu vypíše obsah materializovaného pohledu.