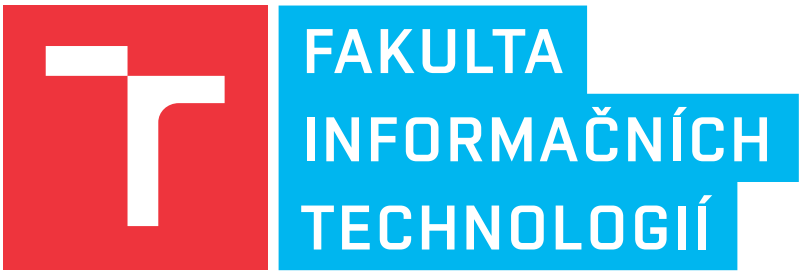
BRNO UNIVERSITY OF TECHNOLOGY



Databázové systémy

Projekt č.: 35

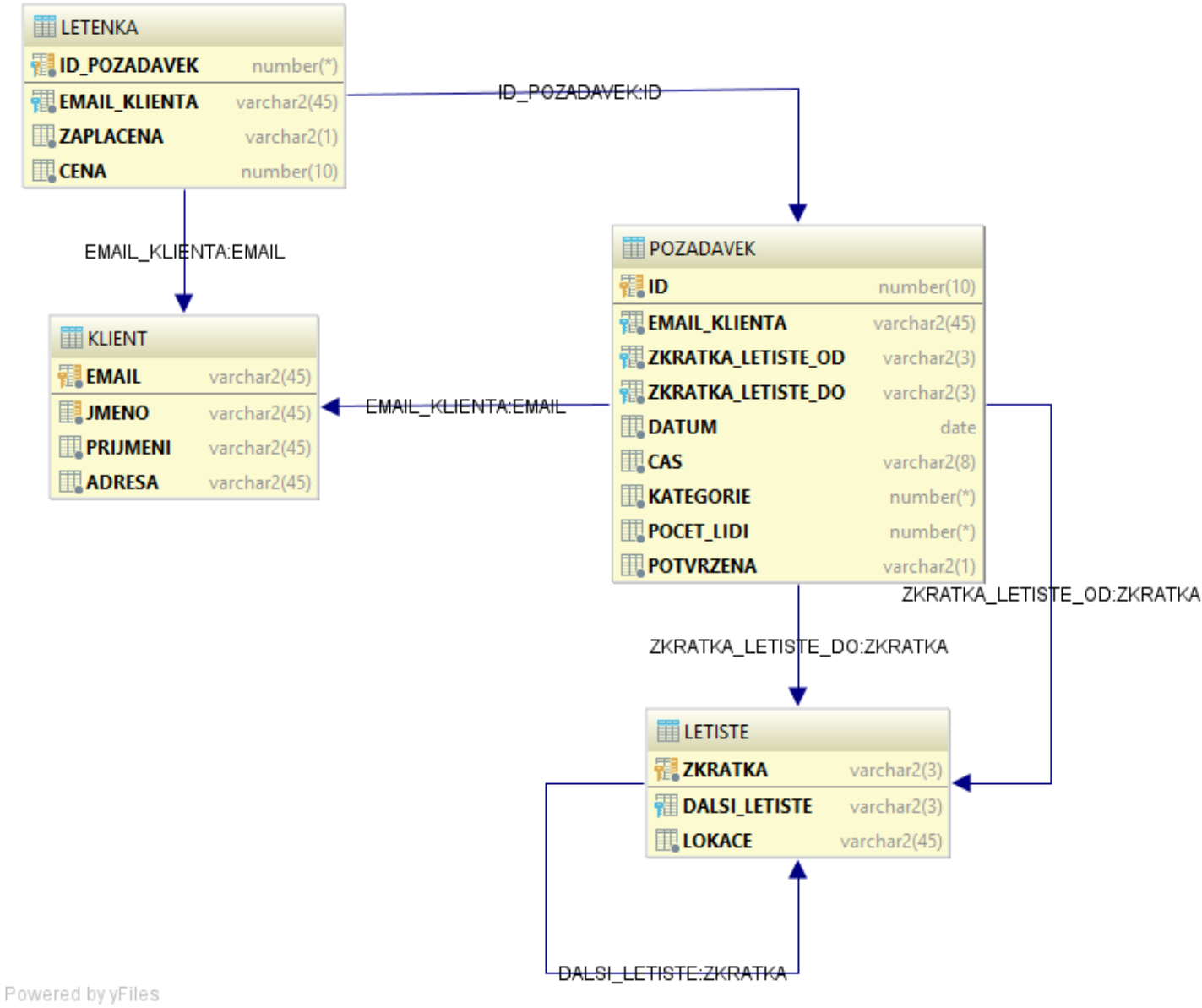
Ivan Manoilov
Farrukh Abdukhalikov

April 29, 2018

1 Zadání

Vaším úkolem je navrhnout webovskou aplikaci pro rezervaci letenek. Systém musí uživateli umožnit požadavek: výchozí a cílové místo, datum, čas, případně některé další parametry (kategorie, letecká společnost apod.)a zobrazí možné spoje, Uživatel zadá adresu el. pošty pro doručení potvrzení objednávky a kontaktní adresu pro doručení letenky po zaplacení.

2 Finální schéma databáze



3 Implementace

SQL skript, který nejprve vytvoří základní objekty schéma databáze a naplní tabulky ukázkovými daty, a poté zadefinuje či vytvoří pokročilá omezení či objekty databáze.

3.1 Triggery

V našem projektu máme implementovano 2 triggery. První trigger je pro kontrolu správností formatu zadaného emailu v tabulce **KLIENT**. Podle nášeho triggeru email *musí* být ve formatu:
`<unikátní název emailu>@<název schránky>.<domena schránky>.`
Druhý trigger je pro auto-generování ID pro primární klíč tabulky **POZADAVEK**. V případě zadání **NULL** v poli primárného klíče tabulky **POZADAVEK**, vygeneruje se nové ID pomocí sekvence **pozadavek_id_seq**.

3.2 Procedury

Stejně jako a triggerů máme 2 procedury. První procedura je zodpovědná za snižení ceny letenky u zákazníků podle adresy. Druha procedura je zodpovědná za smazání požadavků podle zkratky letadla, ona využivá **CURSOR** pro práce z více řadky z tabulky **POZADAVEK**.

3.3 Explain plan a vytvoření indexu

Pro demonstrace optimizace dotazů do databáze pomocí indexů vytvařili jsme dva dotázy a pak prozkoumali jejich výsledky.

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		12	444	10 (10)	00:00:01
1	HASH GROUP BY		12	444	10 (10)	00:00:01
2	MERGE JOIN CARTESIAN		12	444	9 (0)	00:00:01
3	TABLE ACCESS FULL	LETENKA	3	39	3 (0)	00:00:01
4	BUFFER SORT		4	96	7 (15)	00:00:01
5	TABLE ACCESS FULL	KLIENT	4	96	2 (0)	00:00:01

Figure 1: Bez indexu

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		12	444	7 (15)	00:00:01
1	HASH GROUP BY		12	444	7 (15)	00:00:01
2	MERGE JOIN CARTESIAN		12	444	6 (0)	00:00:01
3	TABLE ACCESS FULL	LETENKA	3	39	3 (0)	00:00:01
4	BUFFER SORT		4	96	4 (25)	00:00:01
5	INDEX FAST FULL SCAN	INDEXEXPLAIN	4	96	1 (0)	00:00:01

Figure 2: Po vytvaření indexu

Je vidět, že režie se snizila na 2-3%, a místo přístupu do tabulky optimizováný dotáz jen zkoumá index, což i pro takový nenáročný dotáz(12 položek) dokaže zmenšit využití CPU.

3.4 Materializovaný pohled

Podle požadavků zadání jsme vytvořili taký a **MATERIALIZED VIEW** pro druhého člena týmu. Ten byl vytvořen na zakladů jednoduchého **SELECT** dotazů, a měl atributy **CACHE** pro optimizace načtených dat z **VIEW**, **REFRESH FAST ON COMMIT** pro synchronizace s serverevou databázi, pro něhož musili vytvařet taký a **LOG** před**VIEW**, **BUILD IMMEDIATE** ihned po vytvaření **VIEW** bude naplněn daty a **ENABLE QUERY REWRITE** aby dotázy mohli využivat data z **VIEW**.