A close up of a sign

Description automatically generated

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

IDS

2019/2020

**Dokumentace ke čtvrté části projektu IDS**

Pokorný Matěj - xpokor78

Daniel Andraško – xandra05 Brno, 22. dubna 2020

# Úvod

Tato práce dokumentuje čtvrtou část projektu do předmětu IDS v akademickém roce 2019/2020. Tuto část jsme si rozdělili i kvůli lepší týmové spolupráci do šesti podúkolů, které jsou dokumentovány níže.

# Databázové triggery

Prvním podúkolem této části projektu byla implementace dvou netriviálních databázových triggerů, ze kterých právě jeden musel být pro automatické generování primárních klíčů pro položky v tabulce.

Prvním vytvořeným triggerem je trigger ocisluj\_zpravu, který vygeneruje ID pro každou novou vloženou zprávu. Využívá k tomu sekvenci cislo\_zpravy, která začíná hodnotou 1 a je inkrementována po každém jejím použití o 1.

U ostatních tabulek jsme zachovali automatické generování primárního klíče již implementované ve druhé části projektu, kde jsme využili konstrukce NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY, která je použita již při definici tabulky a je z hlediska jednoduchosti implementace a čitelnosti kódu vhodnější. Generování pomocí triggeru bychom využili ve chvíli kdybychom chtěli mít pro ID nějaké složitější hodnoty, než prosté číslování od 1 s krokem 1.

Druhým vytořeným triggerem je trigger nuluj\_uzivatele, který v rozporu s principy běžných sociálních sítí opravdu zaručuje soukromí uživatele, rozhodne-li se smazat svůj uživatelský účet. Při smazání záznamu z tabulky uzivatel jsou automaticky smazány všechny zprávy, účasti na akcích a označení na fotkách tohoto uživatele. Příspěvky uživatele mazány nejsou, protože ty mohou být sdílené mezi jinými uživateli nebo v soukromých zprávách a jejich náhlé zmizení by mohlo působit chaos ostatním uživatelům sítě.

# Procedury

Podobně jako u triggerů, procedury jsou implementovány dvě. Měly by demonstrovat použití kurzoru, ošetření vyjímek a použití datového typu odkazující na již existující obsah v některé z tabulek.

První implementovanou procedurou je vygeneruj\_uzivatele, která slouží primárně pro testovací účely. Vygeneruje zadaný počet uživatelů, generuje pouze tři různé uživatele, ale pomůže se vyhnout opakovanému kopírování příkazu INSERT. Procedura se volá s parametrem určujícím kolik uživatelů se má generovat, tento počet uživatelů se následně vloží do tabulky uzivatel, konkrétní data se vyberou náhodně ze tří předdefinovaných uživatelů. Tato procedura demonstruje využití vyjímky, která je deklarována je v bloku DECLARE, volána příkazem RAISE a definována před koncem procedury pomoci EXCEPTION. Vyjímka je vyvolána, pokud požadujeme vygenerování méně než jednoho uživatele a vypisuje chybu do debug konzole.

Druhou procedurou je vygeneruj\_spravce, která zkontroluje zda existuje správce John Rambo a pokud ne, vloží ho do tabulky správce. Zde jsou použity kurzory a je zde použito rodné číslo, které demonstruje deklaraci proměnné, která má typ definován již existujícím údajem v tabulce. Pro tento údaj jsme bohužel nenašli praktické použití, ale pro demonstraci použití této konstrukce je příklad dostačující.

# Index pro optimalizaci dotazu

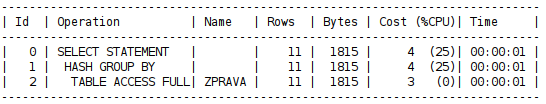
Index pro optimalizaci dotazu, který urychluje jeho vyhledávaní, jsme v našem projektu implementovali jako *ix\_user\_name* pomocou CREATE INDEX.Ten vytvoří bezzhlukový index, co znamená, že uloží data odděleně od řadku v tabulce. Je to v podstatě kopie vybraných stloupcu z dané tabulky.

Následne máme vytvoření dotaz SELECT, který vyhledáva užívatele podle zadaného jména a příjmení a to právě pomocou zmíněného indexu.

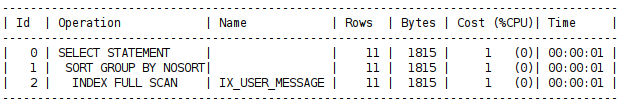
U indexu je podstatné myslet na pořadí stloupcu, kde na prvním míste by měl být takový, který se vyhledáva nejčastěji. V případe, že se tabulka často mění a index se neupravuje, tak je možné, že index vyhledávaní spomalí.

# Explain plan

Již vytvoření EXPLAIN PLAN nám v output tabulce (fotka níže) ukazuje tři postupné operace. Jako první je SELECT STATEMENT, co znamená, že daný plán je vytvoření pro dotaz SELECT. To samé platí i pro druhou operaci a teda vykonávaný plán využíva dotazu GROUP BY. Jako poslední je TABLE ACCESS FULL, což znamená, že se skenuje celá tabulka pro ZPRAVA. A teda každý řádek v ZPRAVA je zpřístupnění a klauzula WHERE sa kontroluje taky pro každý řádek. SELECT dotaz nam vrací počet řádku, které splňují klauzulí WHERE. Dále je možné vidět výkonnostní cenu jednotlivých operací a čas jejich provedení.



Daný dotaz je možné urychlit novým indexem, který indexuje vyhledávané věci v ZPRAVA. Na fotce níže múžeme vidět, že se zvolila metóda procházení indexú, která urychlila proces vyhledávaní a například zmněnšila výkonnostní cenu.



# Přístupová práva pro druhého uživatele

Přístupová práva slouží pro přistup k daným objektúm. V našem projektu mají význam například pro užívatele, který múže prídávat fotky, psát správy apod., neboli admina, který bude mít možnost odstraňovat příspěvky, mazat správy nebo vymazat užívatele.

# Materializovaný pohled pro druhého uživatele

Účelem materializovaného pohledu je uchovat výsledek dotazu, který výchádzi z tabulek nebo jiných pohledú.

Implementovali jsme materializovaný pohled pro užívatele a jejich zprávy. Tedy *uzivatel* a *zprava* jsou master tabulky z kterých čerpáme.

COMMIT je potrvzovaci prikaz, který přidá nový dotaz do pohledu. Pokud COMMIT není volaný, tak sa zmněny neprovedou.