BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2023
http://excel.fit.vutbr.cz

Overenie vlastností SQL kódu

Filip Bali*

Abstrakt
Medzi programátormi je kvalita kódu častý predmet diskusie a kvalitný kód patrí medzi dobré stavebné kamene pre budúci vývoj programu. V súčastnosti je pre takmer každý programovací jazyk rozšírených aj niekoľko programov pre zlepšenie kódu, ktoré je často možné integrovať do vývojového prostredia. Avšak, v prípade jazyku SQL daná ponuka programov mierne absentuje, špeciálne tých s otvoreným kódom. Cieľom diplomovej práce je preto priniesť program s otvoreným kódom, ktorý poskytne možnosť definovať a vkladať vlastné pravidlá pre kontrolu SQL kódu. Zároveň analyzovať možnosti a riešiť problémy kontroly spojené s kontextovou analýzou medzi jednotlivými príkazmi SQL. Tento spôsob kontroly otvára ďalší priestor pre skvalitnenie kódu pred tým, než bude vykonaný v prostredí databázy.

*xbalif00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Riešenie diplomovej práce je motivované myšlienkou zlepšiť kvalitu SQL kódu a to na základe programu, ktorý kontroluje vstupný SQL kód s pomocou definovaných pravidiel. Idea je inšpirovaná už existujúcimi riešeniami pre rôzne iné programovacie jazyky [1][2]. Výhodou by tak mala byť pre programátora väčšia kontrola nad vlastným SQL kódom, ktorá bude dosiahnutá pomocou spätných väzieb ku kódu od spomenutého programu.

Cieľom je sprostredkovať programátorovi spätnú väzbu ku kódu ešte pred jeho vykonaním v prostredí databázy. Rozsah kontroly jednotlivých SQL príkazov sa bude odvíjať od použitých pravidiel, pričom program podporuje jednoduché pridanie aj vlastných pravidiel.

Program zatiaľ funguje v prostredí príkazového riadku, neskôr je možnosť integrácie do vývojových prostredí. Výstupom je zoznam hlásení na štandardnom výstupe, ktoré slúžia k prehľadnému zobrazeniu potenciálnych nezrovnalostí v kóde.

2. Dekompozícia problému a riešenia

Program sa skladá z viacerých celkov, ktoré v rámci programu navzájom spolupracujú. Nasleduje vysvetlenie tých najdôležitejších.

2.1 Abstraktný syntaktický strom

Abstraktný syntaktický strom(ďalej len ako strom) je stromová dátová štruktúra reprezentujúca konkrétny SQL príkaz. Strom vzniká na výstupe syntaktickej analýzy nad daným SQL príkazom a pre jeho získanie je využitá knižnica SQLGlot [3]. Jednotlivé uzly stromu nadobúdajú určitých typov, ktoré reprezentujú daný jazyk [4]. Zloženie stromu jednotlivými typmi uzlov sa tak odvíja od popisovaného SQL príkazu. V rámci plagátu je táto sekcia reprezentovaná v bode číslo jeden.

Program využíva túto stromovú dátovú štruktúru k analýze a kontrole kódu. V rámci kontroly sa prechádza stromom do hĺbky, kde sú na jednotlivé uzly aplikované vybrané pravidlá. Tieto pravidlá reprezentujú konkrétny zámer kontroly ako celok a môžu obsahovať kontrolu aj pre viacero typov uzlov(podrobnejšie v sekcií 2.3). Pravidlo je na uzol aplikované len vtedy, ak obsahuje kontrolu pre daný typ uzla. Kontrola uzla má v rámci každého pravidla pevne daný názov: môže byť definovaná ako prípad, keď je daný uzol navštívený(sufixom *_visit*) alebo ako prípad, keď v rámci prechodu stromom sa opúšťa podstrom, ktorému je daný uzol koreňovým uzlom(sufixom *_leave*).

Výsledný názov kontroly uzla tak predstavuje spojenie názvu uzla s príslušným sufixom typu kontroly. Pre realizáciu tohoto spôsobu kontroly nad stromom využijeme návrhový vzor Návštevník [5, Kapitola 5].

2.2 Štruktúra pre zaznamenávanie databázovej schémy

Aby bolo možné vykonávať aj kontextovú analýzu medzi jednotlivými SQL príkazmi, tak je potrebné zaznamenávať zmeny databázovej schémy. K tomuto účelu je navrhnutá dátová štruktúra, ktorá je vytváraná v pamäti. Cieľom je rozpoznávať príkazy, ktoré majú za následok zmenu databázovej schémy a zaznamenať ekvivalent danej akcie v programe. V rámci plagátu je táto sekcia reprezentovaná v bode číslo dva.

Daná štruktúra zároveň obsahuje rozhranie, cez ktoré je možné nad štruktúrou vykonávať zmeny a overovať aktuálny stav. K tomuto rozhraniu majú prístup aj pravidlá aby mohli overovať analyzovaný príkaz voči aktuálnemu stavu štruktúry a teda v kontexte predchádzajúcich príkazov.

V prípade niektorých príkazov môže program zmeniť jeho usporiadanie a tým ovplyvniť stavbu abstraktného syntaktického stromu. Tento princíp je inšpirovaný spôsobom akým databázové systémy vyhodnocujú príkazy SQL. Napríklad v prípade príkazu *SELECT* je výhodné spracovať klauzule v inom poradí než sú zapísané a to kvôli závislostiam medzi klauzulami [6, Kapitola 1], príklad: kontrola existencie stĺpca v tabuľke je možná až po kontrole existencie samotnej tabuľky.

2.3 Pravidlá

Pravidlo je základným elementom kontroly a predstavuje súbor podmienok pre jednotlivé uzly stromu, ktoré musia byť počas analýzy splnené. Jednotlivé pravidlá môžu byť zamerané na konkrétne typy SQL príkazov alebo na určité úseky kódu. Ich cieľom môže byť kontrola dohodnutého štýlu zápisu kódu, kontrola výkonnostných alebo bezpečnostných vlastností či kontextová kontrola nadväzujúcich SQL príkazov. Špecializovanie pravidiel je možné dosiahnuť pomocou definovaného rozhrania pre pravidlá. Každé pravidlo tak musí mať zhodnú štruktúru s ostatnými pravidlami, ktorej program rozumie.

Rozhranie zároveň umožní aj sprostredkovať výstup užívateľovi. Tvorca nového pravidla má za úlohu vytvoriť správy vo forme popisu nezrovnalosti, ktorú sa pravidlu podarilo nájsť. Zároveň musí mať daná správa identifikátor, na ktorý sa bude pravidlo pri vytvorení správy odkazovať. Následne sa o zvyšnú prácu postará program, ktorý z danej správy vytvorí hlásenie obsahujúce okrem vyššie spomenutého aj pozíciu a ukážku úseku kódu, na ktorom sa kontrola v danom momente vykonávala.

2.4 Hlásenia

Spôsob zobrazenia jednotlivých hlásení je kľúčové pre prácu s programom. K dispozícií je základné a rozšírené zobrazenie. Základné zobrazenie vytvára úsporný prehľad hlásení na výstupe, kde sú dostupné len informácie obsiahnuté v danom hlásení. Zatiaľ čo rozšírené zobrazenie doplňuje to základné o prehlad celého príkazu. Cieľom je aby užívateľ rýchlejšie pochopil ku ktorému SQL príkazu sa daný súbor hlásení vzťahuje.

3. Priebeh a možnosti kontroly

Proces analýzy je znázornený na plagáte v bode číslo tri. Program pred začatím analýzy je schopný získať schému z existujúcej(podporovanej) databázy a ekvivalentne inicializovať vlastnú štruktúru. Následne spracováva vstupné príkazy SQL a podrobuje ich kontrole. Počas kontroly sú nájdené dve nezrovnalosti, ktoré použité pravidlá zistili. Z týchto pravidiel je vytvorené hlásenie, ktoré sa uloží do zoznamu hlásení. Po skončení analýzy sú tieto hlásenia zobrazené užívateľovi aby zvážil prípadnú opravu kódu.

4. Zhrnutie

Táto diplomová práca sa zaoberala tým, že syntaktickú a kontextovú kontrolu kódu na princípe pravidiel je možné uskutočniť aj v prípade príkazov jazyka SQL. K tomuto bol v rámci diplomovej práce implementovaný program, ktorý tieto poznatky využíva a predstavuje realizovateľnosť.

Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu práce RNDr. Marekovi Rychlému Ph.D. za odbornú pomoc a rovnako za jeho čas a trpezlivosť v rámci častých konzultácií.

Literatúra

[1] Nicholas C. Zakas. Eslint - pluggable javascript linter. https://eslint.org/docs/latest/about/. Navštívené: [22.04.2023].

[2] Sylvain Thénault, Claudiu Popa, and contributors. Pylint is a static code analyser for python 2 or 3. https://github.com/pylint-dev/pylint. Navštívené: [22.04.2023].

[3] Mao Toby. Python sql parser and transpiler. https://github.com/tobymao/sqlglot, 2021. Navštívené: [22.04.2023].

[4] Python Software Foundation. Ast - abstract syntax trees. https://docs.python.org/3/library/ast.html. Navštívené: [22.04.2023].

[5] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley, 1994.

[6] Itzik Ben-Gan. *Inside Microsoft SQL Server 2008: T-SQL querying*. Microsoft Press, 2009.