Дисципліна «Технології створення програмних продуктів»

Лекція «Норми кодування (*Coding conventions*) для мови *SQL*»

Викладач: Олександр Блажко, доцент кафедри ІС ОП, blazhko@ieee.org

Мета лекції: отримання знань про норми кодування (*Coding conventions*) для мови *SQL*.

1 Норми кодування для мови програмування *Java*

Мова програмування *SQL* найчастіше використовується я ланка між алгоритмічними мовами програмування, наприклад, мовою *Java*, та СКБД. Тому важливо, щоб рекомендації, представленні у документах з норм кодування (*Code conventions*) для цих мов, пов'язані із форматуванням та визначенням структур даних не протиречили, нормам кодування для *SQL*.

Розглянемо два документи з опису норм кодування мови *Java*:

– норми кодування, розроблені компанією *Sun Microsystems* у 1997 році та оновлені у 1999 році [1];

– норми кодування, розроблені компанією *Google* у 2018 році [2].

Ці документи буде розглянуто лише у розділах, які можуть бути використанні для норм кодуванням мови *SQL*.

1.1 *Sun Microsystems Java Code Conventions*

«Розділ 1. Пукнт 1.1 Мета використання *Code conventions* (*Why Have Code Conventions*)»

Code conventions важливі для програмістів з кількох причин:

– 80% вартості життєвого циклу програмного забезпечення (*software*) йде на його обслуговування (maintenance);

– навряд чи будь-яке програмне забезпечення підтримується протягом усього терміну експлуатації його авторами;

– *Code conventions* покращує читабельність програмного забезпечення, дозволяючи інженерам швидше розуміти новий для них («чужий») код;

– якщо ви оформлюєте свій вихідний код програмного забепечення як програмний продукт, ви повинні переконатися, що він так само добре упакований та оформлений, як і будь-який інший створений вами продукт.

«Розділ 3. Пункт «3.1.1 Початкові коментарі у вихідному файлі (*Beginning Comments*)»

У файлах програмний код повинен починатися з коментаря в стилі мови C (/* */), який містить назву класу, інформацію про версію, дату та повідомлення про авторські права.

«Розділ 4 - Відступи (*Indentation*)»

4-ри прогалини слід використовувати як одиницю відступу. Точна конструкція відступу (прогалини проти табуляції) не визначена. Знаки табуляції повинні бути встановлені точно через кожні 8-м прогалин, а не 4-ри.

«Розділ 4. Пункт 4.1 Довжина рядку (*Line Length*)»

Уникайте рядків з довжиною понад 80 символів, оскільки багато терміналів їх не сприймають. Але приклади для використання в документації повинні мати коротшу довжину рядка, як правило, не більше 70 символів.

«Розділ 4. Пункт 4.2 Обгортання рядків (*Wrapping Lines*)»

Якщо вираз не вміщується в один рядок, розбийте його відповідно до цих загальних принципів:

– перерва рядка після коми;

– перерва рядка перед оператором;

– перевага перервам вищого рівня перед перервами нижчого рівня;

– вирівнювання нового рядок із початком виразу на тому ж рівні попереднього рядка;

Якщо наведені вище правила призводять до заплутаного коду або коду, який тисне на межі зправа, робляться 8-м прогалин.

«Розділ 5 – Коментарі (*Comments*)»

Коментарі слід використовувати для:

– огляду коду та надання додаткової інформації, яка недоступна в самому коді;

– розміщення лише інформації, яка має відношення до осмисленого читання та розуміння програмного коду.

– обговорення нетривіальних або неочевидних проектних рішень, але за уникання дублювання інформації, яка присутня в коді і вже зрозуміла з нього.

Коментарі не повинні перевантажувати програмний код, тому що:

– зайві коментарі надто легко застаріють, тому слід уникати будь-яких коментарів, які можуть застаріти в міру розвитку коду;

– частота коментарів іноді відображає низьку якість коду, коли замість коментування коду краще його переписати, щоб зробити більш зрозумілішим.

Програмний код може бути описано коментарями 4-х стилів: блоковий, однорядковий, розтянутий (*trailing*), прикінцевий.

Блокові коментарі використовуються:

– для надання описів файлів, методів, структур даних і алгоритмів;

– на початку кожного файлу та перед кожним методом.

Блокові коментарі мають наступні особливості форматування:

– використовують обмежуючу пару символів /* */;

– всередині функції або методу повинні мати відступ на той самий рівень, що й код, який вони описують;

– мають передувати порожнім рядком, щоб відокремити їх від решти коду.

Однорядкові коментарі мають наступні особливості:

– коротко описують один рядок програмного коду;

– відображаються в одному рядку з відступом до рівня наступного коду;

– передуються порожнім рядком;

– якщо коментар не можна написати в один рядок, він має відповідати формату блокового коментаря.

Розтянуті коментарі мають наступні особливості:

– використовують обмежуючу пару символів /* */;

– дуже коротко описують рядок програмного коду;

– знаходяться зправа у тому самому рядку програмного коду, який описують, але з відповідним відступом із прогалих для їх візуального відокремлення від коду;

– якщо у фрагменті коду з’являється кілька коротких коментарів, усі вони повинні бути з однаковим відступом.

Прикінцеві коментарі мають наступні особливості:

– використовують символи //;

– можуть закоментувати повний рядок програмного коду або лише його частину.

Розділ «6 –Оголошення (*Declarations*)»

Формат оголошень має наступні рекомендації:

– одне оголошення на рядок;

– оголошення може бути завершено прикінцевим коментарем.

Розділ «7 – Програмні дії (*Statements*)»

Формат опису програмних діймає наступні рекомендації:

проста дія, наприклад, вираз, розміщується в одному рядку;

складна дія, наприклад умована *if, switch* або циклічна *while, for*, використовує символи фігурних дужок для обмеження кордонів {} з наступним розміщенням:

мовна конструкція складної дії *{*

проста дія

..

}

Розділ «8 – Білі прогалини (*White Spaces*)

Візуальне відокремлення різних частин програмного коду реалізовується через так звані білі прогалин (*White Spaces*), які в свою чергу реалізовуються через:

– порожні рядки (*Blank Lines*);

– порожні прогалини (*Blank Spaces*).

Порожні рядки мають наступні особливості:

– покращують читабельність програмного коду, обєднуючи логічно пов’язані частини;

– використовують два порожні рядки між розділами вихідного файлу, між визначеннями класу та інтерфейсу;

– використовують один порожній рядок між методами, між локальними змінними в методі та його першим оператором, перед блоковим або однорядковим коментарем та між логічними розділами всередині методу для покращення читабельності.

Порожні прогалини мають наступні особливості:

– ключове слово, після якого стоїть дужка, має бути відділено прогалиною;

– у списках аргументів методу після коми має бути прогалина;

– усі двійкові оператори повинні бути відокремлені від своїх операндів прогалинами;

– але прогалини не відокремлюють унарні оператори, такі як інкремент ("++") і декремент ("--"), від їхніх операндів;

– вирази в операторі *for* мають бути розділені прогалинами.

Розділ «9 – Норми іменування (Naming Conventions)»

Правила іменування роблять програми більш зрозумілими, полегшуючи їх читання. Вони також можуть надати інформацію про функцію ідентифікатора, наприклад, чи це константа, пакет або клас, що може бути корисним для розуміння коду.

Норми іменування мають наступні рекомендації:

– назви класів мають бути іменниками;

– назви класів повинні мати мінімальну кількість слів але із забезпеченням опису призначення класу;

– якщо назва класу містить декілька слів, тоді слід використовувати цілі слова без акронімів та абревіатур (окрім випадків, коли абревіатура використовується набагато ширше, ніж довга форма, наприклад *URL*-адреса або *HTML*);

– якщо назва класу містить декілька слів, тоді вони пишуться з великою першою літерою кожного слова (так званий *Upper CamelCase*);

– назви методів повинні починатися з дієслова;

– якщо назва методу містить декілька слів, тоді перше слово починається з малої літери та з великою першої літери кожного внутрішнього слова (так званий *lower CamelCase*).

– назви змінних мають бути короткими, але змістовними, вказуючи випадковому спостерігачеві намір їх використання;

– регістри літер назви змінної повторюють умови написання назви методу;

– назви констант мають бути написані великими літерами зі словами, розділеними символами підкреслення ("_").

Розділ «10.5.1 Дужки (*Parentheses*)»

Як правило, добре використовувати круглі дужки у виразах, які включають змішані оператори для уникнення проблем із пріоритетом операторів. Навіть якщо розробнику здається, що пріоритет оператора зрозумілий, інші розробники можуть помилитися.

2 *SQL Code Conventions*

2.1 Joe Celko's SQL Programming Style

В посібнику [3] надано рекомендації, які розроблено з урахуванням змісту книги *Joe Celko's SQL Programming Style*.

Умови найменування:

– назва повинна бути унікальною та не бути зарезервованим ключовим словом;

– назва починається з літери і не може закінчуватися символом підкреслення;

– довжина назви не перевищує 30 знаків:

– назва містить лише латинські літери, цифри та підкреслення для розділення слів;

– назва не може містити скорочення, які не є широко розповсюдженними;

– назва не повинна мати описові префікси, наприклад, *tbl_, col_*, або Угорську нотацію;

– назва таблиці має форму множини;

– назва таблиці не повинна співпадати з назвою її стовпця;

– назва таблиці не повинна містити об'єднання двох імен таблиць разом, щоб створити назву таблиці зв'язків, наприклад, замість назви *cars_mechanics* можна обрати назву *services*;

– назва стовпця має форму однини;

– для назв стовців не використовуйте самостійно назву *id* як основний ідентифікатор для таблиці, а лише як суфікс;

– для обчислюваних даних, наприклад, *SUM*() або *col+100*, використовується псевдонім з ключовим словом *AS*;

– назва збережених процедур має містити дієслово;

– при створенні таблиці закриваюча кругла дужка розміщується на одному рівні зі словом *CREATE*;

– при створенні таблиці опис стовпців починається з відступом у 4-ри прогалини;

– для відступу не використовуються символи табуляції.

Додатково назва стовпця може містити суфікси, які мають універсальне значення, що забезпечує легке читання та розуміння стовпців із коду *SQL*, наприклад:

– _id — унікальний ідентифікатор, наприклад стовпець, який є первинним ключем;

– _status — значення прапорця або інший статус будь-якого типу;

– _total — загальна сума або сума набору значень;

– _num — позначає, що стовпець містить будь-який тип чисел;

– _name — позначає ім`я якоїсь сутності;

– _seq — містить безперервну послідовність значень;

– _date — позначає стовпець, який містить дату;

– _tally — підрахунок якихось значень;

– _size — розмір чогось, наприклад, розміру файлу або одягу;

– _addr — адреса для запису може бути фізичною або нематеріальною;

Додатковою умовою для синтаксису створення таблиці є використання типів даних, загальних для *ANSI SQL*, а не специфічних для СКБД, наприклад:

NUMBER замість *DECIMAL, NUMERIC, INTEGER, BIGINT*;

FLOAT замість *DOUBLE, REAL*;

VARCHAR замість *STRING, TEXT*;

TIMESTAMP замість *DATETIME*.

Умови синтаксису запитів:

– зарезервовані ключові слова використовуються у верхньому регістрі;

– не використовуйте ключові слова, специфічні для СКБД, якщо існує ключове слово *ANSI SQL*, яке виконує такуж функцію;

– ключові слова (*SELECT,FROM,WHERE,INSERT,UPDATE,DELETE)* основного запиту повинні закінчувати на одній межі за рахунок використання прогалин;

– прогалина додається до і після знака “дорівнює” (=);

– прогалина додається після коми (,) та перед наступним виразом;

– новий рядок або прогалина додається перед *AND* або *OR*;

– новий рядок або прогалина додається після кожного визначення ключового слова;

– підзапит вирівнюється по праву сторону з додатковим відступом із закриваючою круглою дужкою на новому рядку в тій самій позиції символу, що й відкриваюча дужка.

2.2 SQL Linter SQLFluff

Серед програм *SQL Linters* відомим є *SQLFluff* - https://github.com/sqlfluff/sqlfluff

Література

1. Code Conventions for the Java Programming Language. URL : https://www.oracle.com/java/technologies/javase/codeconventions-contents.html

2. Google Java Style Guide. URL : https://google.github.io/styleguide/javaguide.html

3. Посібник зі стиль-коду SQL · SQL Style Guide. URL : https://www.sqlstyle.guide/ua/