Bazy danych
przed 10 wykładem

Andrzej Giniewicz

17.05.2024

W aktualnej porcji materiałów uzupełnimy wiedzę o języku SQL o zbiór komend do modyfikacji danych. Dodatkowo poznamy techniczne podstawy łączenia się z bazą z poziomu różnych aplikacji.

Standard języka SQL definiuje cztery osobne języki. Poznaliśmy język SQL DQL (ang. Data Query Language), w skład którego wchodzi tylko komenda `SELECT`. Poznaliśmy również język SQL DDL (ang. Data Definition Language), w skład którego wchodzą trzy komendy `CREATE`, `DROP` oraz `ALTER`. Teraz przyszła pora na SQL DML (ang. Data Manipulation Language), składający się z komend `INSERT`, `UPDATE` oraz `DELETE`. Osoby zainteresowane administracją baz danych mogą we własnym zakresie zapoznać się z podzbiorem języka SQL oznaczanym jako SQL DCL (ang. Data Control Language), który zawiera specyfikację trzech komend — `GRANT`, `REVOKE` oraz `DENY` — służących do nadawania i odbierania uprawnień. Więcej informacji na temat SQL można znaleźć w dokumentacji silnika bazy danych oraz standardzie SQL. Standard SQL ma numer `ISO/IEC 9075`. Normy można wypożyczyć w Punkcie Informacji Normalizacyjnej Politechniki Wrocławskiej[1].

1 Składnia SQL DML

W skład języka SQL DML wchodzą trzy komendy: `INSERT`, `UPDATE` oraz `DELETE`. Zaczniemy od komendy `INSERT`, której jedną wersję mieliśmy już okazję zobaczyć. Dla przypomnienia, aby wynik działania zapytania `SELECT` dodać do innej tabeli, możemy napisać

```
INSERT INTO tabela (kolumny, ...) SELECT ...;
```

Jest to składnia, którą omawialiśmy przy okazji tabel tymczasowych, ponieważ z nimi najczęściej jest używana. Gdy chcemy wstawić do tabeli konkretne wartości niepochodzące z innego zapytania, używamy składni

```
INSERT INTO tabela (kolumny, ...) VALUES
  (wartość_kolumna1, wartość_kolumna2, ...),
  (wartość_kolumna1, ...),
```

[1] Informacje o działaniu PIN znajdują się na stronie `https://biblioteka.pwr.edu.pl/uslugi/normy`.

```
...;
```

Po komendzie VALUES wstawiamy oddzieloną przecinkami listę wstawianych wierszy dowolnej długości. Każdy wiersz musi być tej samej długości, co specyfikacja kolumn po nazwie tabeli. Zamiast wartości istnieje możliwość wpisania DEFAULT, jeśli chcemy wpisać wartość domyślną.

Jeśli wpisujemy wartości dla wszystkich kolumn, możemy pominąć specyfikację kolumn po nazwie tabeli. Wtedy musimy wypisywać wartości w tej samej kolejności, w której tworzona była tabela. Ponieważ może to prowadzić do błędów lub problemów przy zmianie definicji tabeli, zaleca się nie pomijać listy nazw kolumn, nawet jeśli język na to pozwala. Więcej informacji o komendzie INSERT można przeczytać w dokumentacji[2].

Komenda UPDATE wymaga podania warunku oraz zmiany, którą chcemy wykonać. Składnia dyrektywy UPDATE wygląda następująco

```
UPDATE tabela SET
  kolumna=nowa_wartość,
  kolumna=nowa_wartość,
  ...
WHERE warunek;
```

Warto zwrócić uwagę na to, że SQL przegląda tabelę i dla wierszy, które spełniają warunek, oblicza nowe wartości kolumn od lewej do prawej. Dla jednej kolumny najpierw wyliczy nową wartość, następnie wykona podstawienie. W szczególności, jeśli nowa wartość dla kolumny wymienionej jako druga zależy od pierwszej, w obliczeniach zostanie wykorzystana nowa, a nie stara wartość. Przykładowo

```
UPDATE wyniki SET
  punkty=punkty+1,
  suma=punty+bonusy;
```

zwiększy o jeden wszystkie punkty w całej tabeli, a następnie wyliczając sumę, wykorzysta powiększoną o jeden wartość punktów i bonusy. Gdybyśmy zrobili odwrotnie, czyli

```
UPDATE wyniki SET
  suma=punty+bonusy,
  punkty=punkty+1;
```

najpierw wyliczona zostanie suma punktów i bonusów, a dopiero potem punkty zostaną zwiększone o jeden. Oba przypadki powyżej modyfikują całą tabelę, ponieważ nie mają warunku WHERE. Nową wartością wyrażenia może być też wartość domyślna, jeśli użyjemy w jego miejscu słowa kluczowego DEFAULT.

[2] https://mariadb.com/kb/en/insert/.

```
UPDATE users SET
  konto=DEFAULT
WHERE id=32;
```

Zapytanie spowoduje ustawienie domyślnego typu konta dla użytkownika o id równym 32. Więcej informacji o komendzie `UPDATE` można przeczytać w dokumentacji[3].

Ostatnią komendą służącą do modyfikacji danych w tabeli jest `DELETE`. Komenda ta kasuje wiersze spełniające dany warunek.

```
DELETE FROM tabela WHERE warunek;
```

Bardzo ważne, aby nie zapomnieć warunku, ponieważ instrukcja

```
DELETE FROM tabela;
```

jest prawidłową instrukcją SQL, która kasuje wszystkie wiersze z tabeli, pozostawiając ją pustą. Więcej informacji o komendzie `DELETE` można przeczytać w dokumentacji[4].

Przy wszystkich operacjach modyfikujących zawartość tabel musimy uważać, ponieważ wszystkie operacje, szczególnie kasowania, są nieodwracalne, jeśli nie posiadamy kopii zapasowej bazy danych. Z tego powodu regularne tworzenie kopii zapasowych baz jest częstą praktyką. Niemniej jednak nawet częste kopie nie uchronią nas przed utratą choćby kilku wierszy, jeśli przez przypadek użyjemy `DELETE FROM tabela;` bez warunku — utracimy wszystkie zmiany pomiędzy ostatnią kopią a niefortunnym wywołaniem `DELETE`.

2 Architektura łączenia się z bazą

Ponieważ istnieje wiele różnych serwerów baz danych implementujących standard SQL, każdy program korzystający z baz, musiałby być gotowy na wszystkie możliwe interfejsy. Problem ten rozwiązano z wykorzystaniem tak zwanych konektorów. Lista konektorów jest dostępna na stronie każdej bazy danych. W przypadku serwera na zajęcia, jest to `https://mariadb.com/kb/en/connectors/`. Bardzo dużo aplikacji korzysta z jednego z wymienionych konektorów. Do najpopularniejszych należy ODBC, który jest niezależny od platformy i bazy. Każdy serwer baz danych implementujący konektor w standardzie ODBC może na przykład być wykorzystany w różnych aplikacjach, takich jak arkusze kalkulacyjne. Dzięki temu arkusz dodaje obsługę jednego interfejsu, a producenci baz danych dbają o to, aby ich konektory ODBC były aktualne i działały odpowiednio. Innym popularnym konektorem jest JDBC, który jest wykorzystywany przez aplikacje napisane w języku Java. Istnieją również języki, takie jak Python, które oprócz tego, że mają swoje konektory (w przypadku Pythona zgodne z PEP-249[5].), potrafią korzystać z innych, na przykład JDBC i ODBC.

[3] `https://mariadb.com/kb/en/update/`.

[4] `https://mariadb.com/kb/en/delete/`.

[5] https://www.python.org/dev/peps/pep-0249/

Przed przystąpieniem do wyboru konektora dla danego narzędzia, należy zapoznać się z listą obsługiwanych przez narzędzie konektorów, a następnie sprawdzić, które z nich wspiera producent bazy danych. Jeśli mamy kilka opcji, najlepiej w pierwszej kolejności wybrać wersję natywną (czyli PEP-249 dla Pythona, JDBC dla Javy, i tak dalej), a jeśli to niemożliwe, poszukać informacji o wydajności i wsparciu różnych bibliotek dla danych technologii. Wybór rozwiązania popularniejszego zagwarantuje łatwiejsze użytkowanie narzędzi, ponieważ więcej potencjalnych problemów zostało prawdopodobnie już zidentyfikowanych, rozwiązanych i opisanych. Jeśli dany konektor jest nieużywany, należy się zastanowić dlaczego — może być nowy, ale może być też niepraktyczny lub niekompletny. Po wybraniu odpowiednich konektorów musimy pamiętać, że należy je zainstalować w systemie zgodnie z informacjami na stronie `https://mariadb.com/kb/en/connectors/`. Bez tego narzędzia do łączenia się z bazą, nie będą miały możliwości nawiązania połączenia.

Szczegóły łączenia się z poziomu różnych technologii, takich jak pakiet R, język Python, arkusz Execel i inne, są proponowanymi tematami prezentacji, do których zachęcam, jeśli jeszcze nikt ich nie wybrał w grupach laboratoryjnych.