Katedra Mechaniki Stosowanej i Robotyki

SIECI KOMPUTEROWE I BAZY DANYCH

Moduł: Bazy Danych

Temat 6

Wprowadzenie do SQL.
Praca z pojedynczą tabelą

Autor: dr inż. Paweł Penar

Rzeszów 2025

1. Cel laboratorium

Celem laboratorium jest zapoznanie studentów z podstawami języka SQL oraz jego wykorzystanie w pracy z pojedynczą tabelą.

2. Wprowadzenie

SQL to strukturalny język zapytań będący uniwersalnym narzędziem dostępu (tworzenie, wyszukiwania, edycja) do bazy danych. Cechuje go przenośność, standaryzacja oraz prostota.

Polecenia w języku SQL mogą być odczytane jako zdania z języka angielskiego, co gwarantuje jego czytelność. Autorzy [1] wskazują na fakt, że SQL jest językiem deklaratywnym - użytkownik nie określa w jaki sposób ma być realizowany dostęp do danych. Należy zauważyć, że pomimo iż, SQL to standard (LINK), jego implementacja różni się w zależności od producenta bazy danych. Różnice te uwidaczniają się w niewielkich różnicach składni oraz nazwach typów danych.

Na tych laboratoriach będziemy korzystać z SQLite. SQLite to biblioteka w języku C, która implementuje mały, szybki, samodzielny, wysoce niezawodny, w pełni funkcjonalny silnik bazy danych SQL. Jest to najczęściej używany silnik bazy danych na świecie, gdyż jest wbudowany we wszystkie telefony komórkowe i większość komputerów. (LINK)

Baza SQLite definiuje kilka podstawowych typów danych (TEXT, NUMERIC,INTEGER,REAL,NONE) (LINK) które realizują typy wykorzystywane w innych bazach danych LINK. Niektóre z nich podano w poniższej tabeli.

	TYP DANYCH SQL	***OPIS***
TYPY TEKSTOWE **(SQLITE: TEXT)**	**VARCHAR(size)**	Pole tekstowe o zmiennej długości. Od 0 do 255 znaków
	CHAR(size)	Typ tekstowy o długości $0-255$
	TEXT	Typ tekstowy o długości $255 (2^{16}-1)$
	BLOB	Przechowuje dane o zmiennej długości
TYPY LICZBOWE **(SQLITE: NUMERIC, INTEGER,REAL)**	**TINYINT(size)**	Mała liczba rzeczywista o wielkości *size* bitów
	INTEGER; INT	Liczba rzeczywista.
	BIGINT(size)	Duża liczba rzeczywista
	FLOAT(size,d)	Liczba zmiennoprzecinkowa pojedynczej precyzji o całkowitej licznie cyfr o wielkości *size*; w tym d liczb po przecinku.
	DOUBLE(size,d)	Liczba zmiennoprzecinkowa podwójnej precyzji o całkowitej licznie cyfr o wielkości *size*; w tym d liczb po przecinku.
TYPY DATY **(SQLITE: NUMERIC)**	**DATE**	Data
	DATETIME	Kombinacja daty i czasu w zakresie 1000-01-01 00:00:00.000000' to '9999-12-31 23:59:59.999999'

Tworzenie tabeli

Do tworzenie tabeli w SQL służy polecenie *CREATE TABLE*. Jego składnia wymaga podania: nazwy tabeli, nazw kolumn, typów danych oraz innych opcji związanych z kolumnami (np. ograniczenia).

Składnia polecenia *CREATE TABLE* to:

```
CREATE TABLE nazwa_tabeli (nazwa_kolumny_1 typ_danych [opcje],
nazwa_kolumny_2 typ_danych [opcje],
nazwa_kolumny_3 typ_danych [opcje]
…
);
```

gdzie opcje definiują dodatkowe informacje o kolumnie. Niektóre z nich to:

- *PRIMARY KEY* (jest równoważne koniunkcji opcji *UNIQUE* i *NOT NULL*)– mówi o tym że ta kolumna będzie kluczem głównym tabeli. Oznacza to że dane w tej komórce jednoznacznie identyfikują rekord danych. Często jako klucza głównego używa się: numeru pesel, numer indeksu, identyfikatora będącego niepowtarzalną kombinacją liczb i/lub cyfr,
- *AUTOINCREMENT* -opcja dotyczy kolumn o typie liczbowym. Jeśli podczas dodawania nie zostanie podana wartość kolumny, to zostanie tam zapisana wartość o jeden większą niż ta, którą zapisano jako ostatnią. (W bazie SQLite działa dla typu integer, np. *ID integer primary key autoincrement*)
- *NOT NULL* – żadna komórka takiej kolumny nie może być pusta
- *UNIQUE* – dane w takiej kolumnie nie mogą się powtarzać
- *DEFAULT wart*- gdy w tworzonym wierszu pominięto dane które powinny zostać zapisane w tej kolumnie, w kolumnie zapisze się wartość *wart*

Przykład

```
CREATE TABLE Pracownik
( ID INT PRIMARY KEY,
imie varchar(20),
nazwisko varchar(25) );
```

Dodawanie danych do tabeli

W celu dodania rekordów (krotek) do tabeli należy posłużyć się poleceniem *INSERT INTO*

```
INSERT INTO nazwa_tabeli (nazwa_kolumny_1,nazwa_kolumny_2,…) VALUES
(wart1,wart2);
```

Z powyższego schematu wynika że w celu dodania danych do tabeli należy podać jej nazwę, listę kolumn oraz wartości które chcemy podać. Należy pamiętać że kolejność kolumn w liście musi odpowiadać kolejności podanych wartości.

Przykład:

```
INSERT INTO Pracownik (ID,imie, nazwisko) VALUES
(1234,'Jan','Kowalski');
```

Wyszukiwanie danych w tabeli

Jednym z najczęściej używanych poleceń języka SQL jest polecenie SELECT. Polecenie to służy do wyszukiwania danych w tabeli a jego składnia w bazach MySQL jest opisana tutaj: LINK

Uproszczona wersja składnia sporządzona na podstawie [1] ma postać:

```
SELECT wyrażenie FROM nazwa_tabeli [WHERE warunek] [GROUP BY
nazwa_kolumny] [HAVING warunek] [ORDER BY nazwa kolumny ASC|DSC]
```

gdzie

- *wyrażenie* to lista kolumn (ewentualnie wszystkie kolumny oznaczane jako *) i/lub nowa kolumna stworzona poprzez wykonanie funkcji na danych.
- *nazwa_tabeli* – nazwa tabeli w której prowadzimy wyszukiwanie
- *warunek* – warunek wyszukiwania
- *GROUP BY* – warunek grupowania nałożony na kolumnę o nazwie nazwa_kolumny
- *HAVING* – instrukcja nakłada filtr na grupę
- *ORDER BY* – szeregowania danych rosnąco (ASC) lub malejąco (DSC)

Po słowie kluczowym *WHERE* można użyć następujących wyrażeń: *NOT, OR, AND, IN, BETWEEN, LIKE* (wyszukuje wyrażeń podobnych do wzorca) a w wyrażeniu po instrukcji *SELECT* przydatna jest funkcja *COUNT*(nazwa_kolumny) która zwraca liczbę rekordów spełniających dany warunek.

Rozważmy kilka przykładów użycia instrukcji SELECT

a) pokaż wszystkie kolumny tabeli *Pracownik*

```
SELECT * FROM Pracownik
```

b) policz liczbę mężczyzn i kobiet w tabeli *Pracownik*

```
SELECT plec, COUNT(imie) FROM Pracownik GROUP BY plec
```

c) wyszukaj imiona zawierające *aj*

```
SELECT imie, nazwisko, wiem FROM Pracownik WHERE imię like '%aj%'
```

Zadania do wykonania

Uwaga 1: Korzystamy z klienta baz danych o nazwie DbGate. Strona domowa projektu: **https://dbgate.org/**

Uwaga 2: W zadaniach wykorzystujemy bazę plikową SQLite. W szczególności: *employees* do pobrania z serwisu GITHUB*:* *https://github.com/fracpete/employees-db-sqlite*
Do jej rozpakowanie pomocny będzie np. 7-ZIP https://www.7-zip.org

Uwaga 3: Schemat tabeli *employees* z bazy *employess_db* pokazano na rys. 1

Rysunek 1: Tabela employess

1. Zaprojektuj i utwórz tabelę w bazie danych, której nazwę określającą jej zawartość podano w tabeli 1. Tworzona tabela musi zawierać co najmniej 5 kolumn (logicznie powiązanych z tematyką tabeli) a jedna z nich musi być kluczem głównym.
2. Dodaj 5 rekordów do utworzonej bazy danych
3. Wyszukaj nazwisko które zawiera ciąg znaków podany w tabeli 1 pod zadaniami.

   Wskazówka: Dowolny ciąg znaków można zastąpić za pomocą '%'
4. Wyszukaj pracowników których nazwiska zawierają ciąg znaków podany w tabeli 1 pod zadaniami a imiona rozpoczynają się na literę 'e'
5. Wyszukaj pracowników urodzonych przed rokiem podanym w tabeli. Wskazówka: datę zapisz jako '2015-01-01'
6. Wyszukaj pracowników urodzonych w danym roku.
7. Wyszukaj pracowników którzy zostali zatrudnienie w roku X lub w roku Y
8. Policz liczbę mężczyzn i kobiet.
9. Policz liczbę mężczyzn i kobiet których imiona rozpoczynają się na literę podaną w tabeli 1 pod zadaniami.
10. Zaproponuj własne zapytanie do tabeli.

	I	II	III	IV	V	VI
Zad 1	Klient	Faktura	Maszyna	Samochód	Miasto	Komputer
Zad 3	'offre'	'sla'	'eto'	'wer'	'rad'	'brean'
Zad 4	'slaw'	'wer'	'rado'	'rado'	'erto'	'ver'
Zad 5	1955	1962	1959	1956	1953	1964
Zad 6	1957	1962	1956	1955	1952	1964
Zad 7	X:1988 Y:1992	X:1993 Y:1991	X:1989 Y:1994	X:1990 Y:1992	X:1987 Y:1990	X:1990 Y:1999
Zad 9	'A'	'B'	'E'	'R'	'W'	'P'

Tab 1. Dane do zadań z cz. I

Bibliografia

[1] Giergiel J, Giergiel M, K. Kurc, Sieci komputerowe i bazy danych, OWPRz 2010