Język SQL (ang. ***Structured Querty Language***) jest to język zapytań, stosowany w relacyjnych bazach danych do komunikowania się z bazą. Jest również podstawowym językiem programowania baz danych.

Język SQL został opracowany w latach 60. w firmie **IBM**. W 1986 SQL został oficjalnym standardem, wspieranym przez **Międzynarodową Organizację Normalizacyjną (ISO)** i jej członka, **Amerykański Narodowy Instytut Normalizacji (ANSI)**.

Język SQL to **język deklaratywny**, w którym definiuje się warunki, jakie musi spełniać końcowy wynik, natomiast nie definiuje się sposobu, w jaki ten wynik zostanie osiągnięty. W **języku SQL** wyrażenia nazywane są **kwerendami**.

Język SQL wykonuje operacje na **obiektach** tworzących strukturę **hierarchiczną**.
Obiekt w bazie musi mieć niepowtarzalną **nazwę** (ang. ***Alias***).
Odwołanie do obiektu bazy danych:
nazwa_serwera.nazwa_bazy_danych.nazwa_schematu.nazwa_obiektu

Serwer zawiera wiele **baz danych**, baza zawiera wiele **schematów**, a w każdym schemacie może być wiele **tabel**, tabela składa się z wielu **kolumn**.

W praktyce, w ścieżce dostępu do obiektu pomija się elementy powyższego schematu:
- pominiecie **nazwy_serwera** spowoduje wykonanie instrukcji przez serwer z którym jesteśmy połączeni;
- pominięcie **nazwy_bazy_danych** spowoduje wykonanie instrukcji w bazie, z którą jesteśmy połączeni;
- pominięcie **nazwy_schematu** spowoduje wykonywanie instrukcji w domyślnym schamiacie użytkownika **dbo**.

Zmienna relacyjna (**tabela**) posiada dane pogrupowane w **atrybuty** (**kolumny**). **Domeną** (**dziedziną) są** wartości jakie mogą przyjąć dane umieszczone w **kolumnie**.
Identyfikator to nazwa kolumny.

Język SQL podzielony jest na następujące podgrupy:

- **instrukcje DDL** (ang. ***Data Definition Language***) realizują zadania definiowania danych (służą do tworzenia, modyfikowania i usuwania obiektów baz danych):
  - **CREATE**
  - **ALTER**
  - **DROP**
  - **TRUNCATE**

- **instrukcje DML** (ang. ***Data Manipulation Language***) realizują zadania manipulowania danymi (służą do odczytywania i modyfikowania danych):
  - **INSERT**
  - **UPDATE**
  - **DELETE**

- **instrukcje DCL** (ang. ***Data Control Language***) realizują zadania kontrolowania dostępu do danych(służą do nawadnia i odbierania uprawnień użytkownikom):
  - **GRANT**
  - **DENY**

- **REVOKE**

- **instrukcje DQL** (ang. ***Data Querty Language***) realizują zadania pobierania informacji spełniające określone warunki:
  - **SELECT**

- **instrukcje TCL** (ang. ***Transaction Control Language***) realizują zadania sterowania przepływem danych (kontrola transakcji):
  - **COMMIT**
  - **ROLLBACK**
  - **SAVEPOINT**

Składnia języka **SQL**

- identyfikator to nazwa jednoznacznie definiująca **obiekt**, taki jak **baza danych**, **tabela** czy **kolumna**.

Zasady tworzenia identyfikatorów:
- dł. maks. 128 znaków,
- może zawierać cyfry, litery i symbole: **@ $ #** (pozostałe znaki specjalne i spacje są niedozwolone!);
- nie może zaczynać się cyfrą - musi zaczynać się literą (zaczynające się **@** oznaczają zmienną a zaczynające się **#** oznaczają obiekt tymczasowy),
- nie może być słowami kluczowymi języka SQL,
- gdy identyfikator składa się z kilku wyrazów zamiast spacji używamy **_**
- nazwy krótkie, jednoznacznie opisujące obiekt.

Uwaga! *Dla przejrzystości **polecenia języka SQL** piszemy **wielkimi literami**, np.*

CREATE TABLE magazyn;

- literał to wartość stała:
- napisy, ciągi znaków umieszczone w apostrofach, np. 'dzień', 'Numer domu'
- napisy bitowe, poprzedzone literą B, np. B'11011'
- napisy szesnastkowe, poprzedzone literą x, np. X'A18B'
- dokładne wartości liczbowe (ze znakiem lub bez) z możliwością stosowania kropki dziesiętnej, np. -25.4 lub 0.03

UWAGA! *Wszystkie wartości liczbowe, ciągi znaków i daty jeśli nie są identyfikatorami, są traktowane jako stałe, czyli literały, ciągi znaków są umieszczane w apostrofach, np. 'Zakopane', 'BMW','20-09-2014'*

- operator odgrywa rolę łączników, wyróżniamy operatory:

arytmetyczne:
- suma **+**
- równica **-**
- iloczyn *****
- iloraz **/**
- modulo **%**

znakowe:

- konkatencja (złączenie ciągu znaków) **+** lub **||**
- dowolny ciąg znaków **%**
- jeden znak **_**

logiczne:

- koniunkcja **AND**
- alternatywa **OR**
- negacja **NOT**

porównania:

- równy **=**
- mniejszy **<**
- większy **>**
- mniejszy lub równy **<=**
- większy lub równy **>=**
- różny **<>** lub **!=**

operatory relacji:

- **IN (...)** - służy do sprawdzania, czy wartość znajduje się na wcześniej zdefiniowanej liście,
- **BETWEEN ... AND ...** - służy do sprawdzania, czy wartość znajduje się w podanym przedziale,
- **LIKE** - służy do wybierania wartości, które odpowiadają wcześniej ustalonemu wzorcowi. Wzorzec ustalamy za pomoca symboli: % i _
- **REGEXP, RLIKE** - służy do dopasowania do wyrażania regularnego
- **ISNULL** - służy do wyszukiwania wartości **NULL** .

- słowa kluczowe to wyrazy zastrzeżone w **języku SQL**:

- instrukcje, np. **CREATE**, **SELECT**
- klauzule, np. **WHERE**, **JOIN**;
- nazwy typów danych, np. **INT**, **CHAR**;
- nazwy funkcji systemowych, np. **ISNULL**, **ABS**;
- terminy zarezerwowane do późniejszego użycia w systemie.

- komentarze, które są ignorowane przez język SQL:

- komentarz jednowierszowy **--**
- komentarz wielowierszowy **/* ....*/**

- terminatory SQL:

- **terminatory poleceń** kończą polecenia, są związane z dialektami (w Oracle MySQl, PostgreSQL polecenie kończy się średnikiem)
- **terminatory wsadowe** kończą ciąg poleceń.

Typy danych w języku SQL

Typy danych określają rodzaj informacji przechowywanych w kolumnach tabeli.
http://www.promotic.eu/pl/pmdoc/Subsystems/Db/MsSQL/DataTypes.htm

Typy liczbowe

- **liczby całkowite** (ang. ***Integer***)

- **int** - od -2^31 (-2 147 483 648) do 2^31-1 (2 147 483 647)(zajmują 4 bajty pamięci) (może być z unsigned, wtedy jest bez znaku)
- **smallint** - od -2^15 (-32 768) do 2^15-1 (32 767)(zajmują 2 bajty pamięci)
- **bigint** - od -2^63 (-9 223 372 036 854 775 808) do 2^63-1 (9 223 372 036 854 775 807) (zajmują 8 bajtów pamięci)
- **tinyint** - od 0 do 255 (zajmuje 1 bajt pamięci)
- **numeric (p,s)** - precyzja 1-38, domyślna 18 a skala 0-p, domyślna 0) duże wartości liczbowe, np. kwoty pieniężne, np. **numeric(8,2)** będzie przechowywał liczby do 99999999,99
  p - **precyzja** (ang. ***Precision***) liczba miejsc przed przecinkiem - liczba znaczących cyfr, służy do wykonywania obliczeń (musi być dodatnia)
  s - **skala** (ang. ***Scale***) dopuszczalna liczba miejsc po przecinku, nie może być ujemna (równa 0 lub dodatnia)

- **liczby zmiennoprzecinkowe** (ang. ***Floating Point***) (liczby rzeczywiste czyli wartości ułamkowe)

- **real** - (1E-37 do 1E+37 z precyzją przynajmniej 6 znaków po przecinku, zajmuje 4 bajty pamięci)
- **double** - (1E-307 do 1E+308 z precyzją przynajmniej 15 znaków po przecinku, zajmuje 8 bajtów pamięci)
- **float (n)** - typ danych ze zmienną dokładnością, gdzie n wynosi ilość bitów mantysy (1-24, dokładność 7 cyfr, wielkość 4 bajty i 25-53, dokładność 15 czfr i wielkość 8 bajtów)
- **decimal** - podobnie jak numeric

- typy daty i czasu

- **datetime** - typ danych określający datę i czas od 1.1.1753 do 31.12.9999 z dokładnością około 3ms. Wartości są zaokrąglone na .000, .003 a .007. (długość 8 bajtów)
- **smalldatetime** - typ danych określający datę i czas od 1.1.1900 do 6.6.2079 z dokładnością 1m. Wartości do 29.998 są zaokrąglane w dół a wartości od 29.999 są zaokrąglane wgórę na najbliższą minutę (długośc 4 bajty)
- **year –** typ danych określający rok, jeśli zostanie podany zły, jego wartość zmieni się w 0000
- **data –** typ danych określający datę w formacie określonym przez ustawienia serwera

- typy znakowe (ang. ***charakter***)

Uwaga! *Używając typu znakowego w nawiasie określamy **maksymalną liczbę znaków**, którą będzie przechowywał, np. varchar(10), a wprowadzany ciąg znaków umieszczamy w **apostrofach**, np. 'Robert'*

- **char** - łańcuch znaków o stałej długości, długość 8000 znaków
- **varchar** (ang. ***character varying***)- łańcuch znaków o zmiennej długości, długość maksymalna 8000 znaków.

Uwaga! *Jeśli kolumnę definiujemy jako **varchar(10)** system dynamicznie na każdą krotkę przydzieli 10 **znaków**, ale jeśli umieścimy imię 'Maciek' to system przydzieli tylko **6 znaków**.*
*W przypadku **char(10)** i wpisania do krotki 'Maciek' system bez względu na ilość znaków w imieniu przydzieli na nie **10 znaków**.*

- **nchar** - łańcuch znaków Unicode o stałej długości, długość maksymalna 4000 znaków.
- **nvarchar** - łańcuch znaków Unicode o zmiennej długości, długość maksymalna 4000 znaków
- **text** - łańcuch znaków o zmiennej długości, maksymalna długość 2^31-1 (2 147 483 647) znaków.
- **ntext** - łańcuch znaków Unicode o zmiennej długości, długość maksymalna 2^30-1 (1 073 741 823) znaków.

- typ walutowy

- **money** - pieniężny typ danych od -2^63 (-922 337 203 685 477.5808) do 2^63-1 (922 337 203 685 477.5807) z dokładnością jednej dziesięciotysięcznej jednostki (8 bajtów)
- **smallmoney** - pieniężny typ danych od -2^31 (-214 748.3648) do 2^31-1 (214 748.3647) z dokładnością jednej dziesięciotysięcznej (4 bajty)

- typ binarny

- **binary** - dane binarne o stałej długości, długość maksymalna 8000 bajtów.
- **varbinary** - dane binarne o zmiennej długości, długość maksymalna 8000 bajtów.
- typy specjalne
- **image** - dane binarne o zmiennej długości, długość maksymalna 2^31-1 (2 147 483 647) bajtów.
- **xml** -
- **bit** - liczba całkowita 0 lub 1 (rozmiar 1 bit)

- typ pusty NULL

Jest wartością specjalną, oznacza **wartość pustą** (w komórce nie została umieszczona żadna wartość)**. NULL** jest różny od wartości 0 i od pustego ciągu znaków.

- typ wyliczeniowy ENUM

Definicja własnego typu wyliczeniowego

CREATE TYPE tydzien AS ENUM ('PONIEDZIAŁEK', 'WTOREK', 'ŚRODA', 'CZWARTEK', 'PIĄTEK', 'SOBOTA', 'NIEDZIELA');

Uwaga! *Wprowadzone dane rozpoznają wielkości liter: 'Piatek' to nie to samo, co 'piątek'*

Użycie nowego typu

CREATE TABLE przepracowanie_dni (dni tydzien);

Oprogramowanie do tworzenia baz danych:

- MySQL firmy **Oracle**, elementy:
- serwer **MySQL Community Server** http://dev.mysql.com/downloads/mysql/
- śrowisko graficzne **MySQL Workbench** http://dev.mysql.com/downloads/workbench/

- **Pakiety wolnego oprogramowani** do obsługi witryn internetowych, zawierający programy *Open Source*: serwer **Apache**, język skryptowy **PHP**, bazę danych **MySQL** i oprogramowanie uzupełniające:

- **WAMP** (**Windows – Apache – MySQL – PHP**) pakiet dla systemu **Windows**; http://www.wampserver.com/en/
- **LAMP** pakiet dla systemów **Linuksowych**;
- **FAMP** pakiet dla systemu **FreeBSD**;
- **MAMP** pakiet dla systemu **Mac OS X**.

Dialekty języka SQL w zależności od systemu bazodanowego:

- **PL/SQL** (ang. ***Procedural Language/SQL***) serwery Oracle **MySQL Server**
- **PL/pgSQL** (ang. ***Procedural Language/PostgreSQL***) serwery **PostgreSQL**
- **T-SQL** (ang. ***Transact-SQL***) serwery **Microsoft SQL Server** i **Sybase Adaptive Server**
- **SQL PL** (ang. ***SQL Procedural Language***) serwery **IBM**

Instrukcje DDL (ang. *Data Definition Language*)

- **CREATE nazwa_obiektu** - tworzy nowy obiekt;
- **DROP nazwa_obiektu** - usuwa istniejący obiekt;
- **ALTER nazwa_obiektu** - zmienia strukturę obiektu;

Uwaga! *Usunąć bazę może tylko administrator lub właściciel bazy danych, nikt do bazy nie może być w tym czasie podłączony.*

TWORZENIE BAZ DANYCH

Tworzenie nowej bazy danych
CREATE DATABASE nazwa_bazy;
Przykład:
CREATE DATABASE szkola;

Pokazywanie baz danych
SHOW DATABASES;

Użycie bazy danych
USE nazwa_bazy;
Przykład:
USE szkola;

Usuwanie bazy
DROP DATABASE nazwa_bazy;
Przykład:
DROP DATABASE szkola;

TWORZENIE TABEL

Tworzenie tabeli w bazie danych
CREATE TABLE nazwa_tabeli
(nazwa_kolumny_1 typ_kolumny_1 [atrybuty],

nazwa_kolumny_2 typ_kolumny_2 [atrybuty],
...);

Przykład:
CREATE TABLE uczniowie (
id int(11) AUTO_INCREMENT PRIMARY KEY,
imie varchar(30),
nazwisko varchar(40),
pesel varchar(11),
ulica varchar(30),
nr_domu varchar(8),
kod_miasto varchar(6),
miasto varchar(40)
);

*Uwaga! Każda tabela powinna mieć pole, które zawiera **klucz podstawowy PRIMARY KEY**. Kolumna z kluczem głównym nie może być pusta - dodajemy zawsze **NOT NULL***

Pokazywanie tabel w bazie danych
SHOW TABLES;

Pokazywanie pól tabeli
DESCRIBE nazwa_tabeli;
Przykład:
DESCRIBE uczniowie;

Usuwanie tabeli
DROP TABLE nazwa_tabeli;
Przykład:
DROP TABLE uczniowie;

Zasady:
- słowa kluczowe pisane są dużymi literami;
- argumenty wyrażeń pisane są małymi literami;
- nazwa_tabeli i nazwa_kolumny muszą mieć unikatową nazwę;
- każda kolumna musi mieć zdefiniowany typ;
- jeśli kolumna jest typu znakowego trzeba podać jej maks. długość, np. nazwisko varchar(40);
- utworzone tabele są puste;
- bazy danych mają formę katalogów, a tabele są plikami;
- pole to przecięcie wiersza i kolumny;
- wpis do bazy to rekord (krotka).

Atrybuty definiują ograniczenia dla tabeli, określają jakie dane mogą być w niej zapisane.

- **NULL** - w kolumnie mogą wystąpić wartości puste (domyślnie ustawiony atrybut);
- **NOT NULL** - w kolumnie nie mogą wystąpić wartości puste, np. pesel varchar(11) NOT NULL;
- **PRIMARY KEY** – klucz główny, identyfikuje jednoznacznie każdą krotkę (wiersz).

Uwaga! *Co najmniej jedna z kolumn musi być **kluczem głównym**, jest unikatowa i automatycznie indeksowana. Kolumna z kluczem głównym nie może być pusta - dodajemy zawsze **NOT NULL**, np.*
id_goscia int(11) AUTO_INCREMENT PRIMARY KEY NOT NULL;

Jeżeli jest wiele kluczy głównych to podajemy je na końcu, np.
PRIMARY KEY (nazwa_kolumny_1, nazwa_kolumny_2, ...nazwa_kolumny_n);

Klucz główny może zostać wybrany z kluczy kandydujących automatycznie, jeżeli wybieramy go sami to jest to wymuszenie klucza głównego (ang. ***PRIMARY KEY constraint***)

- **AUTO_INCREMENT** - automatyczna numeracja kolumny (automatyczne wypełnianie kolumny po dodaniu kolejnej krotki (wiersza w tabeli);

- **IDENTITY** - oznacza wzrost wartości w kolumnie dla której został zdefiniowany, np.
id_goscia INT IDENTITY (1,1) NOT NULL PRIMARY KEY;

oznacza wzrost wartości kolumny o 1 począwszy od wartości 1

- **DEFAULT** - wprowadzenie do tabeli domyślnej wartości, np.
rok_wydania varchar(4) NOT NULL DEFAULT '2012';

- **UNIQUE** - stosowany, gdy wartości w kolumnie nie mogą się powtarzać, nie blokuje wprowadzenia wartości NULL, np.
tytul varchar(100) NOT NULL UNIQUE;

- **CHECK** pozwala wprowadzić ograniczenia zakresu danych, można używać operatorów NOT, OR, AND, np.
rok_wydania int CHECK (BETWEEN 2010 AND 2014);

MODYFIKACJA TABELI

Modyfikacja tabeli
ALTER TABLE nazwa_tabeli zmiana;

Przykłady:

- dodanie kolumny, np.
ALTER TABLE Ksiazki
ADD liczba_stron varchar(5);

- usunięcie kolumny
ALTER TABLE Ksiazki
DROP COLUMN Liczba_stron;

- zmiana nazwy kolumny
ALTER TABLE nauczyciele
RENAME TO pracownicy;

- zmiana definicji istniejącej kolumny
ALTER TABLE uczniowie MODIFY imie varchar(10) NOT NULL;