

# Podstawowe polecenia SQL dla baz danych MySQL i MariaDB

# Czym jest SQL?

- ▶ Język SQL (ang. *Structured Query Language*) jest językiem używanym do komunikacji użytkownika z system zarządzania bazą danych
- ▶ Obecnie każdy popularny relacyjny system bazodanowy posiada zaimplementowany interpreter języka SQL z zachowaniem pewnego standardu
- ▶ Język SQL umożliwia tworzenie skomplikowanych zapytań do bazy danych w celu otrzymania wyniku

# Za pomocą SQL można:

- ▶ tworzyć: bazy danych, użytkowników, tabele,
- ▶ zarządzać prawami dostępu do baz, do poszczególnych poleceń wydawanych przez użytkowników,
- ▶ zarządzać danymi w tabelach (umieszczać, usuwać, uaktualniać, wyszukiwać),
- ▶ tworzyć funkcje, procedury, wyzwalacze (triggery),
- ▶ i wiele innych...

# SQL można podzielić na następujące grupy poleceń:

SQL DQL (ang. *Data Query Language*)  
- select

SQL DML (ang. *Data Manipulation Language*)  
- update, delete, insert

SQL DDL (ang. *Data Definition Language*)  
- create, drop, alter

SQL TCL (ang. *Transaction Control Language*)  
- begin, commit, start transaction, rollback, savepoint, set transaction isolation level

SQL DCL (ang. *Data Control Language*)  
- grant, revoke

# Krótką historia języka SQL

- ▶ Rozwój relacyjnych systemów bazodanowych - lata 70 XX wieku
- ▶ SEQUEL (ang. *Structured English Query Language*) - pierwszy oficjalny język opracowany przez IBM.
- ▶ Zaimplementowany w 1973 r. w SYSTEM R - przełomowy system bazodanowy powstały jako projekt badawczy IBM
- ▶ Nazwa została zmieniona na SQL z powodu zastrzeżonej nazwy SEQUEL przez jedną z firm brytyjskich
- ▶ Pierwsza standaryzacja SQL:86 - 1986 r.

# Tworzenie i zarządzanie bazami danych. Uprawnienia.

Podstawy baz danych / Artur Niewiarowski



- Tworzenie nowej bazy danych.

```
CREATE DATABASE nazwa;
```

Więcej informacji na stronie internetowej:  
<https://mariadb.com/kb/en/create-database/>

Lista wszystkich baz danych: `show databases;`

- Usunięcie bazy danych.

```
DROP DATABASE nazwa;
```

- Tworzenie nowego użytkownika.

```
CREATE USER 'login'@'localhost' IDENTIFIED BY  
'hasło' PASSWORD EXPIRE;
```

Więcej informacji na stronie internetowej:  
<https://mariadb.com/kb/en/create-user/>

- Usunięcie użytkownika.

```
DROP login@host [, itd];
```



- Uprawnienia użytkowników.

Uprawnienia względem wybranej bazy danych:

```
GRANT [ALL,SELECT,CREATE,...] ON nazwa_bazy.* TO login@host;
```

Uprawnienia globalne:

```
GRANT [ALL,SELECT,CREATE,...] ON *.* TO login@host;
```

**<https://mariadb.com/kb/en/grant/>**

- Uprawnienia użytkowników.

Uprawnienia względem wybranej tabeli w bazie danych:

```
GRANT [ALL,SELECT,INSERT,...] ON nazwa_bazy.tabela TO  
login@host;
```

Uprawnienia względem wybranych kolumn danej tabeli w bazie danych:

```
GRANT [ALL(kolumna1, kolumna2),SELECT(kolumna1,  
kolumna2),INSERT(kolumna),...] ON nazwa_bazy.tabela TO  
login@host;
```

**<https://mariadb.com/kb/en/grant/>**

## ■ Uprawnienia użytkowników.

Odbieranie uprawnień:

```
REVOKE ZDARZENIE ON baza.* from uzytkownik;  
REVOKE ZDARZENIE ON baza.tabela from uzytkownik;  
REVOKE ZDARZENIE (kolumna1, kolumna2,...) ON baza.* FROM  
uzytkownik@host;
```

<https://mariadb.com/kb/en/grant/>

## ■ Uprawnienia - operacje na tabelach systemowych

```
mysql> use mysql;  
Database changed  
mysql> show tables;
```

```
+-----+  
| Tables_in_mysql |  
+-----+  
| columns_priv |  
| db |  
| engine_cost |  
| event |  
| func |  
| general_log |  
| gtid_executed |  
| help_category |  
| help_keyword |  
| help_relation |  
| help_topic |  
| innodb_index_stats |  
| innodb_table_stats |  
| ndb_binlog_index |  
| plugin |  
| proc |  
| procs_priv |  
| proxies_priv |  
| server_cost |  
| servers |  
| slave_master_info |  
| slave_relay_log_info |  
| slave_worker_info |  
| slow_log |  
| tables_priv |  
| time_zone |  
| time_zone_leap_second |  
| time_zone_name |  
| time_zone_transition |  
| time_zone_transition_type |  
| user |  
+-----+
```

## ■ Uprawnienia - operacje na tabelach systemowych

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
Host	char(60)	NO	PRI		
User	char(32)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	
Create_priv	enum('N','Y')	NO		N	
Drop_priv	enum('N','Y')	NO		N	
Reload_priv	enum('N','Y')	NO		N	
authentication_string	text	YES		NULL	
password_expired	enum('N','Y')	NO		N	
password_last_changed	timestamp	YES		NULL	
password_lifetime	smallint(5) unsigned	YES		NULL	
account_locked	enum('N','Y')	NO		N	

## ■ Uprawnienia - operacje na tabelach systemowych

```
mysql> desc db;
```

Field	Type	Null	Key	Default	Extra
Host	char(60)	NO	PRI		
Db	char(64)	NO	PRI		
User	char(32)	NO	PRI		
Select_priv	enum('N','Y')	NO		N	
Insert_priv	enum('N','Y')	NO		N	
Update_priv	enum('N','Y')	NO		N	
Delete_priv	enum('N','Y')	NO		N	
Create_priv	enum('N','Y')	NO		N	
Drop_priv	enum('N','Y')	NO		N	
Grant_priv	enum('N','Y')	NO		N	
References_priv	enum('N','Y')	NO		N	
Index_priv	enum('N','Y')	NO		N	
Alter_priv	enum('N','Y')	NO		N	
Create_tmp_table_priv	enum('N','Y')	NO		N	
Lock_tables_priv	enum('N','Y')	NO		N	
Create_view_priv	enum('N','Y')	NO		N	
Show_view_priv	enum('N','Y')	NO		N	
Create_routine_priv	enum('N','Y')	NO		N	
Alter_routine_priv	enum('N','Y')	NO		N	
Execute_priv	enum('N','Y')	NO		N	
Event_priv	enum('N','Y')	NO		N	
Trigger_priv	enum('N','Y')	NO		N	

## ■ Uprawnienia - operacje na tabelach systemowych

```
mysql> flush privileges;  
Query OK, 0 rows affected (0.00 sec)
```

## ■ Hasła.

Nadawanie i zmiana haseł:

```
SET PASSWORD FOR uzytkownik@host = password('hasło');
```

```
SET PASSWORD = password('hasło');
```

```
ALTER USER 'uzytkownik'@'host' IDENTIFIED BY 'nowe hasło';
```



# information\_schema

<https://dev.mysql.com/doc/refman/8.0/en/information-schema.html>

<https://mariadb.com/kb/en/mariadb/information-schema-tables/>

# information\_schema.engines

```
mysql> select engine, comment, transactions from information_schema.engines;
```

engine	comment	transactions
InnoDB	Supports transactions, row-level locking, and foreign keys	YES
MRG_MYISAM	Collection of identical MyISAM tables	NO
MEMORY	Hash based, stored in memory, useful for temporary tables	NO
BLACKHOLE	/dev/null storage engine (anything you write to it disappears)	NO
MyISAM	MyISAM storage engine	NO
CSV	CSV storage engine	NO
ARCHIVE	Archive storage engine	NO
PERFORMANCE_SCHEMA	Performance Schema	NO
FEDERATED	Federated MySQL storage engine	NULL

<https://mariadb.com/kb/en/choosing-the-right-storage-engine/>

# information\_schema.processlist

```
mysql> select * from information_schema.processlist;
```

ID	USER	HOST	DB	COMMAND	TIME	STATE	INFO
14	designer	localhost:58917	mysql	Sleep	1137		NULL
15	root	localhost:58918	NULL	Sleep	951		NULL
18	root	localhost:58921	tester	Sleep	592		NULL
21	root	localhost:58935	information_schema	Sleep	48		NULL
16	root	localhost:58919	mysql	Sleep	960		NULL
13	designer	localhost:58916	NULL	Sleep	1139		NULL
28	root	localhost:58954	information_schema	Sleep	48		NULL
29	root	localhost:58955	information_schema	Query	0	executing	select * from information_schema.processlist

# information\_schema.routines

```
mysql> select SPECIFIC_NAME, ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_TYPE, DATA_TYPE, IS_DETERMINISTIC, DEFINER from routines
where SPECIFIC_NAME like '%leven%';
```

SPECIFIC_NAME	ROUTINE_SCHEMA	ROUTINE_NAME	ROUTINE_TYPE	DATA_TYPE	IS_DETERMINISTIC	DEFINER
levenshtein	tester	levenshtein	FUNCTION	int	YES	designer@localhost

1 row in set

# performance\_schema

<https://dev.mysql.com/doc/refman/8.0/en/performance-schema.html>

<https://mariadb.com/kb/en/mariadb/performance-schema/>

# performance\_schema.accounts

```
mysql> select * from performance_schema.accounts;
```

USER	HOST	CURRENT_CONNECTIONS	TOTAL_CONNECTIONS
NULL	NULL	25	27
root	localhost	8	38
designer	localhost	2	5

# performance\_schema.hosts

```
mysql> select * from performance_schema.hosts;
```

HOST	CURRENT_CONNECTIONS	TOTAL_CONNECTIONS
NULL	25	27
localhost	8	47

# **Podstawowe polecenia SQL dla MySQL i MariaDB c.d.**



# 1. Tworzenie tabel - polecenie CREATE TABLE

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
    (create_definition,...) [table_options    ]... [partition_opt  
ions]
```

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
    [(create_definition,...)] [table_options    ]... [partition_op  
tions]
```

```
    select_statement
```

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
    { LIKE old_table_name | (LIKE old_table_name) }
```

# 1. Tworzenie tabel - polecenie CREATE TABLE

create\_definition:

```
{ col_name column_definition | index_definition  
| CHECK (expr) }
```

column\_definition:

data\_type

```
[NOT NULL | NULL] [DEFAULT default_value | (expression)]  
[ON UPDATE [NOW | CURRENT_TIMESTAMP] [(precision)]]  
[AUTO_INCREMENT] [ZEROFILL] [UNIQUE [KEY] | [PRIMARY] KEY]  
[INVISIBLE] [{WITH|WITHOUT} SYSTEM VERSIONING]  
[COMMENT 'string']  
[reference_definition]
```

| data\_type

```
AS { (expression) [VIRTUAL | PERSISTENT | STORED] } }  
[UNIQUE [KEY]] [COMMENT 'string']
```

constraint\_definition:

```
CONSTRAINT [constraint_name]
```

# 1. Tworzenie tabel - polecenie CREATE TABLE

Przykład:

```
CREATE TABLE `Klienci` (  
  ID_klient int unsigned NOT NULL AUTO_INCREMENT,  
  imie varchar(70) NOT NULL,  
  nazwisko varchar(250) NOT NULL,  
  PESEL varchar(11) DEFAULT 'brak',  
  rok_urodzenia date NOT NULL,  
  PRIMARY KEY (ID_klient),  
  UNIQUE KEY uniq_Klienci (imie,nazwisko,PESEL) USING BTREE  
)
```

# 1. Tworzenie tabel - polecenie CREATE TABLE

Przykład:

```
CREATE TABLE Pliki (  
  ID_pliki int(10) unsigned NOT NULL AUTO_INCREMENT,  
  ID_klient int(10) unsigned DEFAULT NULL,  
  nazwa_pliku varchar(100) NOT NULL,  
  plik blob,  
  ts_pliki timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
  CURRENT_TIMESTAMP,  
  PRIMARY KEY (`ID_pliki`),  
  KEY `Klienci_pliki` (`ID_klient`) USING BTREE,  
  CONSTRAINT `Klienci_pliki` FOREIGN KEY (`ID_klient`) REFERENCES  
  `Klienci` (`ID_klient`) ON UPDATE CASCADE  
)
```

# 1. Tworzenie tabel - polecenie CREATE TABLE

## Przykład:

```
mysql> create table uzytkownicy1 select * from uzytkownicy where 0;  
Query OK, 0 rows affected (0.02 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc uzytkownicy;
```

Field	Type	Null	Key	Default	Extra
ID_uzytkownicy	int(11)	NO	PRI	NULL	auto_increment
login	varchar(20)	NO		NULL	
haslo	varchar(255)	NO		NULL	
imie	varchar(20)	YES		NULL	
nazwisko	varchar(200)	YES		NULL	

```
5 rows in set (0.03 sec)
```

```
mysql> desc uzytkownicy1;
```

Field	Type	Null	Key	Default	Extra
ID_uzytkownicy	int(11)	NO		0	
login	varchar(20)	NO		NULL	
haslo	varchar(255)	NO		NULL	
imie	varchar(20)	YES		NULL	
nazwisko	varchar(200)	YES		NULL	

```
Podstawy baz danych / Artur Niewiarowski  
5 rows in set (0.02 sec)
```

# 1. Tworzenie tabel - polecenie CREATE TABLE

## Przykład:

```
mysql> create table uzytkownicy2 like uzytkownicy;  
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> desc uzytkownicy;
```

Field	Type	Null	Key	Default	Extra
ID_uzytkownicy	int(11)	NO	PRI	NULL	auto_increment
login	varchar(20)	NO		NULL	
haslo	varchar(255)	NO		NULL	
imie	varchar(20)	YES		NULL	
nazwisko	varchar(200)	YES		NULL	

```
5 rows in set (0.04 sec)
```

```
mysql> desc uzytkownicy2;
```

Field	Type	Null	Key	Default	Extra
ID_uzytkownicy	int(11)	NO	PRI	NULL	auto_increment
login	varchar(20)	NO		NULL	
haslo	varchar(255)	NO		NULL	
imie	varchar(20)	YES		NULL	
nazwisko	varchar(200)	YES		NULL	

```
5 rows in set (0.04 sec)
```

Bez referencji!!!

# 1. Tworzenie tabel - polecenie CREATE TABLE

Przykład:

```
CREATE TABLE trojkaty (  
  ID_trojkaty bigint(20) unsigned NOT NULL DEFAULT uuid_short(),  
  a int(11) DEFAULT NULL,  
  `b` int(11) DEFAULT NULL,  
  `c` int(11) DEFAULT NULL,  
  `obwod` int(11) DEFAULT (`a` + `b` + `c`),  
  PRIMARY KEY (`ID_trojkaty`)  
)
```

Begin Transaction   Text   Filter   Sort					
	ID_trojkaty	a	b	c	obwod
▶	66727513833490	1	2	3	6
	66727513833491	2	3	4	9

# 1. Tworzenie tabel - polecenie CREATE TABLE

Przykład:

```
CREATE TABLE `trojkaty2` (  
  `ID_trojkaty` varchar(36) NOT NULL DEFAULT uuid(),  
  `a` double DEFAULT NULL,  
  `b` double DEFAULT NULL,  
  `c` double DEFAULT NULL,  
  `obwod` double GENERATED ALWAYS AS (`a` + `b` + `c`) VIRTUAL,  
  PRIMARY KEY (`ID_trojkaty`) USING BTREE  
);
```

Begin Transaction   Text   Filter   Sort   Import   Export					
ID_trojkaty	a	b	c	obwod	
▶ c3cf04c7-8661-11eb-8d5e-e0d55eadd5db	1	2	3	6	
f63c509b-8661-11eb-8d5e-e0d55eadd5db	2	3	4	9	



# 1. Tworzenie tabel - polecenie CREATE TABLE

## Przykład:

```
mysql> update trojkaty2 set obwod=100;  
1906 -
```

The value specified for generated column 'obwod' in table 'trojkaty2' has been ignored

```
mysql> select * from trojkaty2;
```

ID_trojkaty	a	b	c	obwod
c3cf04c7-8661-11eb-8d5e-e0d55eadd5db	1	2	3	6
f63c509b-8661-11eb-8d5e-e0d55eadd5db	2	3	4	9

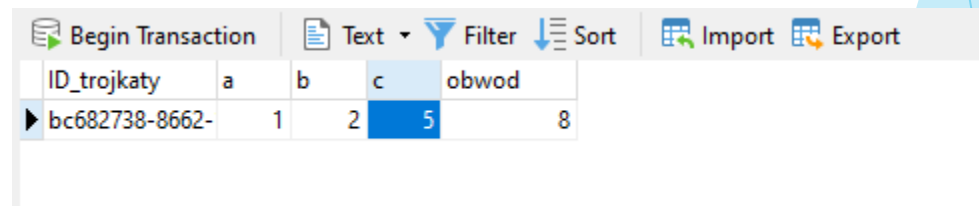
```
2 rows in set (0.02 sec)
```

```
mysql>
```

# 1. Tworzenie tabel - polecenie CREATE TABLE

Przykład:

```
CREATE TABLE `trojkaty4` (  
  `ID_trojkaty` varchar(36) NOT NULL DEFAULT uuid(),  
  `a` double DEFAULT NULL,  
  `b` double DEFAULT NULL,  
  `c` double DEFAULT NULL,  
  `obwod` double GENERATED ALWAYS AS (`a` + `b` + `c`) STORED,  
  PRIMARY KEY (`ID_trojkaty`) USING BTREE  
) ;
```



The screenshot shows a database management interface with a table named 'trojkaty4'. The table has five columns: 'ID\_trojkaty', 'a', 'b', 'c', and 'obwod'. The 'c' column is highlighted in blue. The first row of data contains the values: 'bc682738-8662-', 1, 2, 5, and 8. The interface includes a toolbar with buttons for 'Begin Transaction', 'Text', 'Filter', 'Sort', 'Import', and 'Export'.

ID_trojkaty	a	b	c	obwod
bc682738-8662-	1	2	5	8

# 1. Tworzenie tabel - polecenie CREATE TABLE

## Przykład:

```
mysql> update trojkaty4 set obwod=100;
```

```
1906 -
```

The value specified for generated column 'obwod' in table 'trojkaty4' has been ignored

```
mysql> select * from trojkaty4;
```

ID_trojkaty	a	b	c	obwod
bc682738-8662-11eb-8d5e-e0d55eadd5db	1	2	5	8

```
1 row in set (0.01 sec)
```

```
mysql>
```

<https://mariadb.com/kb/en/generated-columns/>

# 1. Tworzenie tabel - polecenie CREATE TABLE

## Przykład:

```
mysql> create table kwadrat (ID_kwadrat int primary key auto_increment,  
A double check(A>0));  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> desc kwadrat;
```

Field	Type	Null	Key	Default	Extra
ID_kwadrat	int(11)	NO	PRI	NULL	auto_increment
A	double	YES		NULL	

2 rows in set (0.02 sec)

```
mysql> insert into kwadrat (A) values (5.5);  
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into kwadrat (A) values (-5.5);  
4025 - CONSTRAINT `kwadrat.A` failed for `baza_dla_aniewiarowski`.`kwadrat`  
mysql> select * from kwadrat;
```

ID_kwadrat	A
1	5.5

1 row in set (0.01 sec)

Podstawy baz danych / Artur Niewiarowski

```
mysql>
```

# Wybrane typy danych w MySQL/MariaDB

<https://mariadb.com/kb/en/library/data-types/>

<https://dev.mysql.com/doc/refman/8.0/en/data-types.html>

# Wybrane typy danych w MySQL/MariaDB

## Ciągowe typy danych:

- **char**
- **varchar**
- **blob** (przechowuje wartości binarne, np. pliki)
- **text** (przechowuje duże teksty), jak również (tinytext, mediumtext, longtext)
- **enum** (umieszczenie jednej z podanych wartości w enum)
- **set** (umieszczenie wielu wartości podanych w set)
  
- **json** (MySQL - <https://dev.mysql.com/doc/refman/8.0/en/json.html>)
- **json** (MariaDB - <https://mariadb.com/kb/en/json-data-type>)

# Wybrane typy danych w MySQL/MariaDB

## Liczbowe typy danych:

- **bool** (przechowuje wartości: 0 lub 1)
- **decimal(M,D)** (M - wartość całkowita: od 1 do 65 znaków, D - wartość po przecinku: od 0 do 30 znaków)
- **smallint [unsigned]** (liczba całkowita od -128 do 127 znaków)
- **int [unsigned]** (-2147483648-2147483647 [0-4294967295])
- **bigint [unsigned]**
- **float [unsigned]**
- **double [unsigned]**

# Wybrane typy danych w MySQL/MariaDB

## Typy danych związane z czasem i datą:

- **datetime** (przechowuje datę i czas, rok z przedziału: od 1000 do 9999) - format: np. '2023-03-11 12:54:00'
- **date** (przechowuje wyłącznie datę)
- **timestamp** (przechowuje datę i czas)
- **time** (przechowuje wyłącznie czas)
- **year** (przechowuje wyłącznie rok)



## 2. Usuwanie tabel - polecenie DROP TABLE

Składnia polecenia:

```
DROP TABLE [IF EXISTS] nazwa_tabeli;
```

Przykład:

```
DROP TABLE Klienci;
```

### 3. Zmiana struktury tabel - polecenie ALTER TABLE

Składnia polecenia zmieniającego kolumnę:

```
ALTER TABLE nazwa_tabeli CHANGE nazwa_kolumny  
nowa_nazwa_kolumny nowe parametry dla kolumny;
```

Przykład:

```
ALTER TABLE Klienci CHANGE imie imiona varchar(50) not null;
```

## 3. Zmiana struktury tabel - polecenie ALTER TABLE

Składnia polecenia dodającego kolumnę:

```
ALTER TABLE nazwa_tabeli ADD nazwa_kolumny  
parametry dla kolumny;
```

Przykład:

```
ALTER TABLE Klienci ADD drugie_imie varchar(12) default ' brak ';
```

## 3. Zmiana struktury tabel - polecenie ALTER TABLE

Składnia polecenia usuwającego kolumnę:

```
ALTER TABLE nazwa_tabeli DROP nazwa_kolumny;
```

Przykład:

```
ALTER TABLE Klienci DROP drugie_imie;
```

### 3. Zmiana struktury tabel - polecenie **ALTER TABLE**

Składnia polecenia zmieniającego nazwę tabeli:

```
ALTER TABLE stara_nazwa RENAME nowa_nazwa;
```

Przykład:

```
ALTER TABLE student RENAME student2;
```

### 3. Zmiana struktury tabel - polecenie ALTER TABLE

Składnia polecenia zmieniającego kolejność kolumn:

```
ALTER TABLE `nazwa_tabeli`  
MODIFY COLUMN `kolX` varchar(100) NOT NULL FIRST,  
MODIFY COLUMN `ID` int(10) UNSIGNED NOT NULL  
AUTO_INCREMENT AFTER `kolX`;
```

Przykład:

```
ALTER TABLE `autor`  
MODIFY COLUMN `nazwa` varchar(100) NOT NULL FIRST,  
MODIFY COLUMN `ID_autor` int(10) UNSIGNED NOT NULL  
AUTO_INCREMENT AFTER `nazwa`;
```

## 4. Umieszczanie danych w tabeli - polecenie INSERT INTO

Składnia polecenia:

```
INSERT INTO nazwa_tabeli (kolumna_1, kolumna_2, ... )  
VALUES ('wartość_1', 'wartość_2');
```

Przykład:

```
INSERT INTO Klienci  
(imie, nazwisko, PESEL, data_urodzenia)  
VALUES ('Jan', 'Kowalski', '89020212345', '1989-02-02');
```

## 4. Umieszczanie danych w tabeli - polecenie INSERT INTO

Składnia polecenia:

```
INSERT INTO nazwa_tabeli  
VALUES ('wartość_1', 'wartość_2');
```

Przykład:

```
INSERT INTO Klienci VALUES ('Jan', 'Kowalski',  
'89020212345', '1989-02-02');
```

```
Insert into Klienci values ('Jan', 'Kowalski', '89020212345',  
'1989-02-02'),  
( 'Jan2', 'Kowalski2', '99020212345', '1989-02-02'),  
( 'Jan3', 'Kowalski3', '19020212345', '1989-02-02' );
```



## 4. Umieszczanie danych w tabeli - polecenie INSERT INTO

Przykład:

```
mysql> insert into uzytkownicy select * from tmp_uzytkownicy;  
Query OK, 3 rows affected  
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> insert into uzytkownicy (login, imie,  
nazwisko) select username, name, surname from tmp_uzytkownicy;  
Query OK, 3 rows affected  
Records: 3 Duplicates: 0 Warnings: 0
```

## 5. Usuwanie danych z tabeli - polecenie DELETE FROM

Składnia polecenia:

```
DELETE FROM nazwa_tabeli WHERE warunek;
```

Przykład:

```
DELETE FROM Klienci WHERE Nazwisko = 'Kowalski' AND Imie  
= 'Jan' or Nazwisko like 'Nowak%' AND Imie  
like 'J__' limit 10;
```

```
DELETE FROM Klienci WHERE ID_klienci='5';
```

## 5. Usuwanie danych z tabeli - polecenie DELETE FROM

Składnia polecenia:

```
DELETE FROM nazwa_tabeli WHERE warunek;
```

Przykład:

```
delete from uzytkownicy where ID_typu in  
(select ID_typu from typu where nazwa_typu='root');
```

# 5. Podmiana danych w tabeli - polecenie REPLACE

Składnia polecenia:

```
REPLACE INTO nazwa_tabeli ...
```

Przykład:

```
mysql> replace into uzytkownicy (ID_uzytkownicy, login, haslo) values (1, 'jkowalski2', password(uuid(
))), (2, 'anowak2', password(uuid()));
Query OK, 4 rows affected (0.00 sec)
Records: 2 Duplicates: 2 Warnings: 0
```

```
mysql> select * from uzytkownicy;
```

ID_uzytkownicy	login	haslo	imie	nazwisko	ID_typ
1	jkowalski2	*93735F191CC618F552C38EF5A62B59B858CA0A97	NULL	NULL	NULL
2	anowak2	*FFD3117EA31C27BCB5F922CE535679718CF2F537	NULL	NULL	NULL

2 rows in set (0.02 sec)

## 6. Uaktualnianie danych w tabeli - polecenie UPDATE

Składnia polecenia:

```
UPDATE nazwa_tabeli SET nazwa_kolumny_1 = '5' ,  
nazwa_kolumny_2 = 'wartość' WHERE warunek;
```

Przykład:

```
UPDATE Klienci SET imie = 'Michał' WHERE Nazwisko =  
'Kowalski' and Imie = 'Jan';
```

## 7. Odczytywanie danych z tabeli - polecenie SELECT

Składnia polecenia:

```
SELECT nazwa_kolumny_1, nazwa_kolumny_2, ... FROM  
nazwa_tabeli WHERE warunek
```

Przykład:

```
SELECT imie, nazwisko, PESEL FROM Klienci WHERE Nazwisko =  
'Kowalski' and Imie = 'Jan';
```

## 7. Odczytywanie danych z tabeli - polecenie SELECT

Składnia polecenia:

```
SELECT * FROM nazwa_tabeli WHERE warunek;
```

Gwiazdka po *select* oznacza uwzględnienie w wyniku wszystkich widocznych kolumn

Przykład:

```
SELECT * FROM Klienci WHERE Nazwisko = 'Kowalski' and  
(Imie = 'Jan' or Imie = 'Janek');
```

## 7. Pozostałe polecenia SQL w SZBD MySQL/MariaDB

- **SHOW DATABASES** - wyświetla listę baz danych, do których ma dostęp użytkownik
- **SHOW TABLES** - wyświetla listę zawierającą nazwy wszystkich tabel i widoków w aktualnej bazie danych
- **DESCRIBE (lub DESC) nazwa\_tabeli** - opisuje szczegółowo wybraną tabelę
- **USE nazwa\_bazy** - przełącza się do innej bazy danych
- **SHOW CREATE TABLE nazwa\_tabeli** - wyświetla kod SQL tworzący daną tabelę
- **EXIT** - zamknięcie terminala/wylogowanie z bazy danych



# Operatory i funkcje matematyczne (MySQL /MariaDB)

# Operatory i funkcje matematyczne

```
root@db-it:/home/artur.niew X + v
MariaDB [(none)]> help sin
Name: 'SIN'
Description:
Syntax
-----
SIN(X)

Description
-----
Returns the sine of X, where X is given in radians.

Examples
-----
SELECT SIN(1.5707963267948966);
+-----+
| SIN(1.5707963267948966) |
+-----+
| 1 |
+-----+

SELECT SIN(PI());
+-----+
| SIN(PI()) |
+-----+
```

Wbudowany manual - polecenie *help*

# Operatory matematyczne

- ▶ + (dodawanie)
- ▶ - (odejmowanie)
- ▶ \* (mnożenie)
- ▶ / (dzielenie)
- ▶ % (dzielenie modulo)

# Operatory i funkcje matematyczne

```
mariadb> select 2+8+10;
```

```
+-----+  
| 2+8+10 |  
+-----+  
|      20 |  
+-----+
```

```
1 row in set (0.02 sec)
```

```
mariadb> select 2-8-10, 45-45;
```

```
+-----+-----+  
| 2-8-10 | 45-45 |  
+-----+-----+  
|     -16 |      0 |  
+-----+-----+
```

```
1 row in set (0.03 sec)
```

```
mariadb> select 3*5 as wynik1, 1.5*2 wynik2;
```

```
+-----+-----+  
| wynik1 | wynik2 |  
+-----+-----+  
|      15 | 3.0    |  
+-----+-----+
```

```
1 row in set (0.03 sec)
```

```
mariadb> select 1/2, 0.5/2;
```

```
+-----+-----+  
| 1/2    | 0.5/2   |  
+-----+-----+  
| 0.5000 | 0.25000 |  
+-----+-----+
```

```
1 row in set (0.03 sec)
```

```
mariadb> select 4%2, 5 mod 2;
```

```
+-----+-----+  
| 4%2 | 5 mod 2 |  
+-----+-----+  
|    0 |        1 |  
+-----+-----+
```

```
1 row in set (0.04 sec)
```

# Operatory i funkcje matematyczne

```
mariadb> desc liczby;
```

Field	Type	Null	Key	Default	Extra
A	int(11)	YES		NULL	
B	double	YES		NULL	
C	decimal(10,2)	YES		NULL	

```
mariadb> select A, B, C, A+B*C + 100 / 2 as wynik from liczby;
```

A	B	C	wynik
1	2	3.00	57
5	5.6	4.57	80.592
0	-1.6	-10.10	66.16

3 rows in set (0.03 sec)

# Funkcje matematyczne

- ▶ **abs()** (oblicza wartość absolutną)
- ▶ **sin(), cos(), tan(), cot()** (funkcje sinus, cosinus, itd.)
- ▶ **pow()** (funkcja potęgowa)
- ▶ **sqrt()** (obliczanie pierwiastka)
- ▶ **pi()** (zwraca wartość liczby pi, 3.14159265)
- ▶ **rand()** (zwraca wartość losową)

# Funkcje matematyczne

- ▶ **log()**, **ln()**, **log2()**, **log10()** (obliczanie logarytmów)
- ▶ **degrees()** (funkcja konwertująca radiany na stopnie)
- ▶ **radians()** (funkcja konwertująca stopnie na radiany)
- ▶ **mod()** (funkcja modulo)
- ▶ **round()** (zaokrąglanie wartości liczbowych)
- ▶ **crc32()** (cykliczny kod nadmiarowy)

# Operatory i funkcje matematyczne

```
mariadb> select tan(0), tan(pi()), cot(1), cot(pi()/2);
```

tan(0)	tan(pi())	cot(1)	cot(pi()/2)
0	-1.2246467991473532e-16	0.6420926159343306	6.123233995736766e-17

```
mariadb> select pow(2,3), round(pow(27,1/3));
```

pow(2,3)	round(pow(27,1/3))
8	3

```
mariadb> select pow(2,3), round(pow(27,1/3)), sqrt(4);
```

pow(2,3)	round(pow(27,1/3))	sqrt(4)
8	3	2

```
mariadb> select abs(-5), abs(5), abs(-2*8);
```

abs(-5)	abs(5)	abs(-2*8)
5	5	16

```
mariadb> select sin(0), sin(pi()/2), cos(0), cos(pi());
```

sin(0)	sin(pi()/2)	cos(0)	cos(pi())
0	1	1	-1



# Operatory i funkcje matematyczne

```
mariadb> select rand(), rand(), rand()*10-5;
```

rand()	rand()	rand()*10-5
0.0014161393059567914	0.18863082229032258	4.389057242673875

```
mariadb> select log(0), log(10,100);
```

log(0)	log(10,100)
NULL	2

```
mariadb> select degrees(1.5707963267948966), radians(90);
```

degrees(1.5707963267948966)	radians(90)
90	1.5707963267948966

# Operatory i funkcje matematyczne

```
mariadb> desc liczby;
```

Field	Type	Null	Key	Default	Extra
A	int(11)	YES		NULL	
B	double	YES		NULL	
C	decimal(10,2)	YES		NULL	

```
mariadb> select A, B, C, round( sin(A)+cos(B)*abs(C) + 100 / 2, 2) as wynik  
from liczby;
```

A	B	C	wynik
1	2	3.00	49.59
5	5.6	4.57	52.59
0	-1.6	-10.10	49.71

# Operatory i wybrane funkcje komparacji (MySQL/Mar iaDB)

# Operatory i wybrane funkcje komparacji

- ▶ **between** wartość1 and wartość2 (wartość pomiędzy)
- ▶ wartość1 = wartość2 (porównanie dwóch wartości)
- ▶ >, <, >=, <= (znaki: większości, mniejszości, ...)
- ▶ <>, != (znaki różności)
- ▶ **is**, **is not** (wyrażenie: jeżeli, jeżeli nie)
- ▶ wartość1 **like** '%wartość\_2%' (podobny)
- ▶ wartość1 **like** '\\%wartość\\_2%'
- ▶ **in()** (zawiera się w ...)

# Operatory i wybrane funkcje komparacji

```
mariadb> select * from liczby;
```

ID_liczby	A	B	C
1	1	2.5	NULL
2	3	6.0009	2.23
3	-1	-100	-34.56

```
mariadb> select * from liczby where B between 1 and 10 and sin(B) between 0 and A;
```

ID_liczby	A	B	C
1	1	2.5	NULL

```
mariadb> select * from liczby where B >= 1 and B <= 10 and sin(B) >= 0 and sin(B) <= A;
```

ID_liczby	A	B	C
1	1	2.5	NULL

```
mariadb> select * from liczby where B = 2.5 or A != -1 or A <> 3;
```

ID_liczby	A	B	C
1	1	2.5	NULL
2	3	6.0009	2.23
3	-1	-100	-34.56

# Operatory i wybrane funkcje komparacji

```
mariadb> select * from liczby where C = null;  
Empty set
```

```
mariadb> select * from liczby where C is null;
```

ID_liczby	A	B	C
1	1	2.5	NULL

```
mariadb> select * from liczby where C is not null;
```

ID_liczby	A	B	C
2	3	6.0009	2.23
3	-1	-100	-34.56

```
mariadb> select * from liczby where not C is null;
```

ID_liczby	A	B	C
2	3	6.0009	2.23
3	-1	-100	-34.56

```
mariadb> select *, isnull(A), isnull(B), isnull(C) from liczby where isnull(C) and not isnull(A) and isnull(B) = 0;
```

ID_liczby	A	B	C	isnull(A)	isnull(B)	isnull(C)
1	1	2.5	NULL	0	0	1

Dla *null* nie  
stosujemy  
znaku: „=”

# Operatory i wybrane funkcje komparacji

```
mariadb> select * from liczby;
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

```
mariadb> select * from liczby where komentarz like 'liczby';  
Empty set
```

```
mariadb> select * from liczby where komentarz like '%liczby%';
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby

```
mariadb> select * from liczby where komentarz like '%liczby';
```

ID_liczby	A	B	C	data	komentarz
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby

# Operatory i wybrane funkcje komparacji

```
mariadb> select * from liczby;
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

```
mariadb> select * from liczby where komentarz like '%\null\';
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'

```
mariadb> select * from liczby where data like '%-02-%';
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

```
mariadb> select * from liczby where B like '._.';
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'

Maskowanie znaków

Możliwość operacji na datach

Możliwość operacji na liczbach



# Operatory i wybrane funkcje komparacji

```
mariadb> select * from liczby;
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

```
mariadb> select * from liczby where komentarz like '\%wartości ujemne%';
```

ID_liczby	A	B	C	data	komentarz
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

```
1 row in set (0.04 sec)
```

Maskowanie  
znaków

```
mariadb> select * from liczby where komentarz like 'wartości ujemne%';
```

```
Empty set
```

```
mariadb> select data like '%-02-%', data from liczby;
```

data like '%-02-%'	data
1	2022-02-02 12:00:15
0	2023-01-02 15:10:15
1	2025-02-02 00:00:00

Możliwość  
umieszczenia  
*like* po *select*

# Operatory i wybrane funkcje komparacji

```
mariadb> select * from liczby where A in (1,3,5,10);
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby

```
mariadb> select * from liczby where A in (5,10, 100);
```

Empty set

```
mariadb> select * from liczby where not A in (5,10, 100) and A not in (5,10, 100);
```

ID_liczby	A	B	C	data	komentarz
1	1	2.5	NULL	2022-02-02 12:00:15	dwie liczby i 'null'
2	3	6.0009	2.23	2023-01-02 15:10:15	trzy liczby
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

```
mariadb> select * from liczby where not A in (1,3,5,10);
```

ID_liczby	A	B	C	data	komentarz
3	-1	-100	-34.56	2025-02-02 00:00:00	%wartości ujemne%

# Funkcje agregujące (MySQL/ MariaDB)

# Funkcje agregujące

Funkcje operują na danych w kolumnie. W nawiasach może zostać podana wyłącznie jedna nazwa kolumny lub zbiór kolumn będący formułą, np.  $\text{kol1} + \text{kol2} * \text{kol3} / \text{kol4}$ . Jeżeli zostanie podana wartość liczbowa, to zostanie powielona.

- **avg()** (funkcja obliczająca średnią)
- **count()** (funkcja zliczająca liczbę wierszy)
- **min()** (funkcja zwracająca wartość minimalną)
- **max()** (funkcja zwracająca wartość maksymalną)
- **sum()** (funkcja sumująca wartości)

# Funkcje agregujące

Zestaw danych:

```
create table tab(liczba double);  
insert into tab values (-1), (1), (2), (3), (4.1), (5.8), (8), (10);
```

Przykłady:

Wartość średnia

```
mysql> select avg(liczba) from tab union all  
select avg(liczba) from tab where liczba>=0;  
+-----+  
| avg(liczba) |  
+-----+  
|          4.1125 |  
| 4.8428571428571425 |  
+-----+
```

Zliczanie wierszy

```
mysql> select count(*) from tab union all select  
count(liczba) from tab where liczba>=0;  
+-----+  
| count(*) |  
+-----+  
|          8 |  
|          7 |  
+-----+
```

Funkcja zwracająca wartość minimalną

```
mysql> select min(liczba) from tab union all  
select min(liczba) from tab where liczba>=0;  
+-----+  
| min(liczba) |  
+-----+  
|          -1 |  
|           1 |  
+-----+
```

Funkcja zwracająca wartość maksymalną

```
mysql> select max(liczba) from tab union all  
select max(liczba) from tab where liczba<0;  
+-----+  
| max(liczba) |  
+-----+  
|          10 |  
|          -1 |  
+-----+
```

Funkcja sumująca

```
mysql> select sum(liczba) from tab union all select sum(liczba) from tab where liczba<0;  
+-----+  
| sum(liczba) |  
+-----+  
|          32.9 |  
|          -1 |  
+-----+
```

# Funkcje agregujące - polecenia: group by i having

Zestaw danych:

```
mysql> select * from student;
```

ID_student	imie	nazwisko
31	Jan	Kowalski
32	Jan	Muszka
33	Janina	Nowakowska
34	Stefania	Nowak
35	Janina	Buła

# Funkcje agregujące - polecenia: group by i having

Przykład zastosowania polecenia: group by

```
mysql> select * from student group by imie;
```

ID_student	imie	nazwisko
31	Jan	Kowalski
33	Janina	Nowakowska
34	Stefania	Nowak

Efekt: rekordy pogrupowane względem kolumny imie, zawierające unikalne wartości dla imie, część rekordów została usunięta z wyniku zapytania. Taki efekt można otrzymać m.in. w MySQL/MariaDB. Są systemy bazodanowe, gdzie po *group by* należy wymienić wszystkie kolumny wchodzące w skład zapytania (tj. po select).

# Funkcje agregujące - polecenia: group by i having

Przykład zastosowania polecenia: group by w połączeniu z funkcją agregującą count()

```
mysql> select imie, count(*) ile_takich_samych_imion from  
student group by imie;
```

imie	ile_takich_samych_imion
Jan	2
Janina	2
Stefania	1

```
3 rows in set
```



# Funkcje agregujące - polecenia: group by i having

Przykład zastosowania polecenia: group by w połączeniu z funkcją agregującą count()

```
mysql> select imie, count(*) ile_takich_samych_imion from  
student group by imie having ile_takich_samych_imion>1;
```

```
+-----+-----+  
| imie   | ile_takich_samych_imion |  
+-----+-----+  
| Jan    | 2 |  
| Janina | 2 |  
+-----+-----+  
2 rows in set
```

Wyświetlenie wyłącznie tych imion, które powtarzają się min. 2 razy. Polecenie *having* używamy tak jak *where*, ale do filtracji wyniku zapytania po poleceniu *group by*.

*Przyp.* *Where* używamy zaraz po *from* i przed *group by*. Polecenie *group by* będzie grupowało te rekordy, które zostaną przefiltrowane przez *where*.

# Funkcje agregujące - polecenia: group by i having

Przykład zastosowania polecenia: group by w połączeniu z funkcją agregującą count() - przykład operacji na liczbach

```
mariadb> desc liczby;
```

Field	Type	Null	Key	Default	Extra
A	int(11)	YES		NULL	
B	double	YES		NULL	
C	decimal(10,2)	YES		NULL	

A	B	C
1	2	3,00
5	5,6	4,57
0	-1,6	-10,10
2	10	10,00
4	3	-3,40

# Funkcje agregujące - polecenia: group by i having

Przykład zastosowania polecenia: group by w połączeniu z funkcją agregującą count() - przykład operacji na liczbach

Ile liczb jest parzystych w kolumnie A?

```
mariadb> select A, A mod 2 as wynik_modulo from liczby;
```

A	wynik_modulo
1	1
5	1
0	0
2	0
4	0

```
mariadb> select count(A) ile_parzystych, A mod 2 as  
wynik_modulo from liczby group by wynik_modulo having  
wynik_modulo = 0;
```

ile_parzystych	wynik_modulo
3	0

A	B	C
1	2	3,00
5	5,6	4,57
0	-1,6	-10,10
2	10	10,00
4	3	-3,40

# Funkcje agregujące - polecenia: group by i having

Reasumując

```
SELECT
    count(*) ile,
    kol2
FROM
    tabela
WHERE          --where filtruje rekordy do zgrupowania
    kol1 > 5    --where umieszcza się przed group by
GROUP BY
    kol2
HAVING        --having filtruje rekordy będące wynikiem grupowania
    ile > 10;  --having umieszcza się po group by
```

# Wybrane funkcje operujące na ciągach tekstowych (MySQL /MariaDB)

# Wybrane funkcje operujące na ciągach

```
root@db-it:/home/artur.niew X + v
MariaDB [(none)]> help left
Name: 'LEFT'
Description:
Syntax
-----
LEFT(str,len)

Description
-----
Returns the leftmost len characters from the string str, or
NULL if
any argument is NULL.

Examples
-----
SELECT LEFT('MariaDB', 5);
+-----+
| LEFT('MariaDB', 5) |
+-----+
| Maria |
+-----+
```

Wbudowany manual - polecenie *help*

# Wybrane funkcje operujące na ciągach tekstowych

- ▶ **concat()** (łączy znaki w ciąg)
- ▶ **hex()** (oblicza wartość heksalną na podstawie zadanego parametru)
- ▶ **instr()** (szuka jednego ciągu w drugim)
- ▶ **lcase()**, **lower()** (zamieniają duże litery na małe)
- ▶ **left()** (wycina znaki od lewej strony ciągu)
- ▶ **length()** (zwraca długość podanego ciągu)
- ▶ **ltrim()** (usuwa spacje po lewej stronie zadanego ciągu)
- ▶ **mid()**, **substring()** (wycina środek ciągu)
- ▶ **position()**, **locate()** (zwraca pozycję ciągu w zadanym ciągu)

# Wybrane funkcje operujące na ciągach tekstowych

- ▶ **repeat()** (powtarza n razy podany ciąg)
- ▶ **replace()** (zamienia ciąg na inny w zadanym ciągu)
- ▶ **reverse()** (zamienia kolejność liter w ciągu)
- ▶ **right()** (wycina znaki od prawej strony ciągu)
- ▶ **rtrim()** (usuwa spacje po prawej stronie zadanego ciągu)
- ▶ **soundex()**, ... **sounds like** ... (zwraca w postaci ciągu brzmienie w języku angielskim podanego wyrazu)
- ▶ **space()** (tworzy spacje)
- ▶ **trim()** (usuwa po obu stronach ciągu spacje)
- ▶ **ucase()**, **upper()** (zamienia małe na duże litery w zadanym ciągu)



# Wybrane funkcje operujące na ciągach

```
mariadb> select instr('Bazy danych', 'danych'), instr('Bazy danych', 'bazy');
```

instr('Bazy danych', 'danych')	instr('Bazy danych', 'bazy')
6	1

```
mariadb> select instr('Bazy danych', 'danych'), instr('Bazy danych', 'bazy');
```

instr('Bazy danych', 'danych')	instr('Bazy danych', 'bazy')
6	1

```
mariadb> select concat ('a','b','c',1,2,3);
```

concat ('a','b','c',1,2,3)
abc123

```
mariadb> select left('Katedra Informatyki', 8);
```

left('Katedra Informatyki', 8)
Katedra

# Wybrane funkcje operujące na ciągach

```
mariadb> select * from liczby;
```

A	B	C
1	2	3.00
5	5.6	4.57
0	-1.6	-10.10
2	10	10.00
4	3	-3.40

```
mariadb> select *, concat('A=',A,', ', 'B=', B, ', C=', C) as tekst from liczby;
```

A	B	C	tekst
1	2	3.00	A=1, B=2, C=3.00
5	5.6	4.57	A=5, B=5.6, C=4.57
0	-1.6	-10.10	A=0, B=-1.6, C=-10.10
2	10	10.00	A=2, B=10, C=10.00
4	3	-3.40	A=4, B=3, C=-3.40

# Wybrane funkcje operujące na ciągach

```
mariadb> select length('Politechnika Krakowska'), length('żółć'), char_length('żółć');
```

length('Politechnika Krakowska')	length('żółć')	char_length('żółć')
22	8	4

```
mariadb> select length('To jest "cytat"'), length('To jest \'cytat\''), length("To jest 'cytat'");
```

length('To jest "cytat"')	length('To jest \'cytat\''')	length("To jest 'cytat'")
15	15	15

```
mariadb> select length('To jest \n"cytat"');
```

length('To jest \n"cytat"')
16

```
mariadb> select lower('To jest \n"cytat"');
```

lower('To jest \n"cytat"')
to jest "cytat"

# Wybrane funkcje operujące na ciągach

```
mariadb> select reverse('ABC');
```

reverse('ABC')
CBA

```
mariadb> select upper('it'), lower('IT'), ucase('it'), lcase('IT');
```

upper('it')	lower('IT')	ucase('it')	lcase('IT')
IT	it	IT	it

aliasy:

```
mariadb> select upper('it') a, lower('IT') b, ucase('it') as c, lcase('IT') as d;
```

a	b	c	d
IT	it	IT	it

# Wybrane funkcje operujące na ciągach

```
mariadb> select ltrim('   IT   ') a, rtrim('   IT   ') b, trim('   IT   ') c;
```

a	b	c
IT	IT	IT

```
mariadb> select substring('Katedra Informatyki', 1, 7),  
substr('Katedra Informatyki', 1, 7), mid('Katedra Informatyki', 1, 7);
```

substring('Katedra Informatyki', 1, 7)	substr('Katedra Informatyki', 1, 7)	mid('Katedra Informatyki', 1, 7)
Katedra	Katedra	Katedra

```
mariadb> select replace('Instytut Informatyki', 'instytut', "Katedra");
```

replace('Instytut Informatyki', 'instytut', "Katedra")
Katedra Informatyki

```
mariadb> select locate('Katedra', 'Katedra Informatyki'), position("katedra" in "Katedra Informatyki");
```

locate('Katedra', 'Katedra Informatyki')	position("katedra" in "Katedra Informatyki")
1	1

```
mariadb> select repeat('Katedra ', 3);
```

repeat('Katedra ', 3)
Katedra Katedra Katedra

# Wybrane funkcje operujące na ciągach

```
mariadb> select soundex('Word'), soundex('World'), 'world' sounds like 'word';
```

soundex('Word')	soundex('World')	'world' sounds like 'word'
W630	W643	0

```
mariadb> select soundex('man'), soundex('men'), 'man' sounds like 'men';
```

soundex('man')	soundex('men')	'man' sounds like 'men'
M000	M000	1

# Wybrane funkcje daty i czasu (MySQL /MariaDB)

# Wybrane funkcje daty i czasu

- ▶ **adddate()** (zwraca datę utworzoną poprzez dodanie lub odjęcie dni)
- ▶ **addtime()** (zwraca czas utworzony poprzez dodanie lub odjęcie czasu)
- ▶ **curdate()** (zwraca aktualną datę)
- ▶ **curtime()** (zwraca aktualny czas)
- ▶ **date()** (zwraca datę z podanego ciągu)
- ▶ **datediff()** (oblicza różnicę dat i zwraca ją w dniach)
- ▶ **now()** (zwraca aktualną datę i czas)
- ▶ **day(), month(), year()** (na podstawie ciągu zwracają odpowiednio: dzień, miesiąc, rok)
- ▶ **dayname()** (zwraca nazwę dnia w j. angielskim na podstawie daty)



# Wybrane funkcje daty i czasu

- ▶ `dayofmonth()`, `dayofweek()`, `dayofyear()` (na podstawie daty zwracają odpowiednio: numer dnia w miesiącu, dnia w tygodniu i dnia w roku)
- ▶ `hour()`, `minute()`, `second()` (na podstawie daty zwracają odpowiednio godzinę, minuty, sekundy)
- ▶ `monthname()` (nazwa miesiąca podanej daty)
- ▶ `sysdate()` (zwraca aktualną datę i czas w momencie wywołania funkcji)
- ▶ `sec_to_time()` (zamienia sekundy na czas)
- ▶ `time()` (zwraca czas z podanej daty i czasu)
- ▶ `timediff()` (oblicza różnicę czasów)
- ▶ `time_to_sec()` (zamienia czas na sekundy)

# Wybrane funkcje daty i czasu

```
mariadb> select adddate('2030-01-01', 365);
```

```
+-----+  
| adddate('2030-01-01', 365) |  
+-----+  
| 2031-01-01                |  
+-----+
```

```
mariadb> select curdate(), curtime();
```

```
+-----+-----+  
| curdate() | curtime() |  
+-----+-----+  
| 2023-08-01 | 13:57:39 |  
+-----+-----+
```

```
mariadb> select now(), dayname(now());
```

```
+-----+-----+  
| now()      | dayname(now()) |  
+-----+-----+  
| 2023-08-01 13:58:23 | Tuesday        |  
+-----+-----+
```

```
mariadb> select addtime('14:00:00', '01:10:15');
```

```
+-----+  
| addtime('14:00:00', '01:10:15') |  
+-----+  
| 15:10:15                          |  
+-----+
```

```
mariadb> select now(), date(now());
```

```
+-----+-----+  
| now()      | date(now()) |  
+-----+-----+  
| 2023-08-01 14:00:09 | 2023-08-01 |  
+-----+-----+
```

```
mariadb> select time_to_sec('00:02:05');
```

```
+-----+  
| time_to_sec('00:02:05') |  
+-----+  
| 125                      |  
+-----+
```

# Wybrane funkcje daty i czasu

```
mariadb> select day(now()), month(now()), year(now());
```

day(now())	month(now())	year(now())
1	8	2023

```
mariadb> select dayofmonth(now()), dayofweek(now()), dayofyear(now());
```

dayofmonth(now())	dayofweek(now())	dayofyear(now())
1	3	213

```
mariadb> select hour(now()), minute(now()), second(now()), now();
```

hour(now())	minute(now())	second(now())	now()
14	57	2	2023-08-01 14:57:02

```
mariadb> select monthname(now()), time(now());
```

monthname(now())	time(now())
August	14:57:14

# Wybrane funkcje daty i czasu

```
mariadb> select sysdate(), sleep(2), sysdate();
```

sysdate()	sleep(2)	sysdate()
2023-08-01 15:02:32	0	2023-08-01 15:02:34

1 row in set (2.04 sec)

```
mariadb> select now(), sleep(2), now();
```

now()	sleep(2)	now()
2023-08-01 15:02:48	0	2023-08-01 15:02:48

1 row in set (2.04 sec)

```
mariadb> select curtime(), timediff(curtime(), '12:00:00');
```

curtime()	timediff(curtime(), '12:00:00')
15:05:15	03:05:15

```
mariadb> select curdate(), datediff(curdate(), '2021-01-01');
```

curdate()	datediff(curdate(), '2021-01-01')
2023-08-01	942