**Logowanie do bazy danych:**mysql -u root -p

**Tworzenie i usuwanie – instrukcje CREATE i DROP**

CREATE --> tworzy nowy obiekt

CREATE DATABASE nowa\_baza; --> tworzenie nowej bazy danych o nazwie nowa\_baza

DROP --> usuwa istniejący obiekt

DROP DATABASE nowa\_baza; --> usuwanie bazy danych o nazwie nowa\_baza

**Wyświetlanie baz danych**

SHOW DATABASES; --> wyświetlanie baz danych utworzonych na serwerze

USE nazwa; - podłączenie się do bazy

SHOW TABLES; - wyświetlenie tabel w bazie danych

DESCRIBE nazwa\_tabeli --> zostanie wyświetlona struktura tabeli

**Atrybuty kolumn**

PRIMARY KEY --> klucz podstawowy

AUTO\_INCREMENT --> definiowanie automatycznej inkrementacji

DEFAULT --> jest stosowany do wprowadzania do kolumny wartości domyślnej

NOT NULL --> w danej kolumnie nie mogą się znaleźć wartości puste

UNIQUE --> unikatowe wartości w kolumnie, np. PESEL nie może się powtórzyć wśród wielu pracowników

Przykład tworzenia tabeli osoba7

* CREATE TABLE osoba7 (id int AUTO\_INCREMENT NOT NULL PRIMARY KEY, nazwisko varchar(60), imie varchar(40) DEFAULT ‘Marek’);

**Wstawianie rekordów do bazy danych**

INSERT --> służy do wstawiania nowych wierszy do tabeli

* INSERT INTO osoba7 (imie, nazwisko) VALUES (‘Adam’, ‘Kowalski’);;

**Zapytanie SELECT**

Wyświetlanie wszystkich rekordów

* SELECT \* FROM nazwa\_tabeli;

Wyświetlamy wszystkich o imieniu Adam

* SELECT \* FROM pracownik WHERE imie=’Adam’;

Wyświetlamy wszystkich których imie **zaczyna** się na literę A.

* SELECT \* FROM pracownik WHERE imie LIKE ‘%A’;

Wyświetlamy wszystkich których imie **kończy** się na literę A.

* SELECT \* FROM pracownik WHERE imie LIKE ‘A%’;

Wyświetlamy wszystkich którzy zawierają ski w nazwisku

* SELECT \* FROM pracownik WHERE nazwisko LIKE ‘%ski%’;

Wyświetlamy wszystkie rekordy gdzie pole telefon jest puste

* SELECT \* FROM pracownik WHERE telefon IS NULL;

Wyświetlamy wszystkie rekordy gdzie pole telefon **nie jest** puste i imię to Adam.

* SELECT \* FROM pracownik WHERE telefon IS NOT NULL AND imie=’Adam’;

**Instrukcja UPDATE**

Służy do aktualizowania danych. W instrukcji UPDATE podajemy nazwę tabeli a w klauzuli SET nazwę modyfikowanej kolumny oraz przypisaną jej nową wartość.

* UPDATE nazwa\_tabeli SET imie=’Robert’;

Jeżeli modyfikacja będzie dotyczyła wybranych wierszy do instrukcji należy dołączyć klauzulę WHERE.

* UPDATE nazwa\_tabeli SET imie=’Krzysztof’ WHERE nazwisko=’Polak’;

**Instrukcja DELETE**

Usunięcie wszystkich wierszy z wybranej tabeli

* DELETE FROM nazwa\_tabeli;

Po dodaniu klauzuli WHERE z tabeli zostaną usunięte wiersze, które spełniają określony w klauzuli warunek.

* DELETE FROM nazwa\_tabeli WHERE kod\_pocztowy=’43-400’;

**Instrukcja ALTER TABLE**

Zmiana struktury tabeli może polegać na usunięciu lub dodaniu kolumny, dodaniu atrybutu lub usunięcia atrybutu.

Tabele modyfikujemy za pomocą instrukcji ALTER TABLE.

Dodawanie kolumny do tabeli

* ALTER TABLE nazwa\_tabeli ADD COLUMN nazwa\_kolumny varchar(15);

Usunięcie kolumny z tabeli

* ALTER TABLE nazwa\_tabeli DROP COLUMN nazwa\_kolumny;

Modyfikowanie kolumny

* ALTER TABLE nazwa\_tabeli MODIFY COLUMN nazwa\_kolumny varchar(60);

Zmiana nazwy tabeli

* ALTER TABLE nazwa\_tabeli RENAME TO nowa\_nazwa;

**Kopiowanie tabeli**

Kopiowanie tabeli 1 do tabeli 2

* CREATE TABLE tabela2 SELECT \* FROM tabela1;

**Odtworzenie struktury tabeli**

* SHOW CREATE TABLE nazwa\_tabeli;

**Znaki wieloznaczne**

Znak % pasuje do dowolnych znaków, jedynym wyjątkiem jest wartość NULL.  
Wyświetlanie wszystkich rekordów, oprócz wartości NULL

* SELECT \* FROM nazwa\_tabeli WHERE nazwa\_kolumny LIKE ‘%’;

Wyświetlamy wszystkie rekordy, które rozpoczynają się od litery c i kończy na literę y

* SELECT \* FROM nazwa\_tabeli WHERE nazwa\_kolumny LIKE ‘c%y’;

**Znak podkreślenia** oznacza dokładnie jedno wystąpienie dowolnego znaku i używa się go podobnie jak znaku %.

**Sortowanie wyników**

W celu posortowania wyników należy użyć klauzuli ORDER BY

ASC – porządek rosnący  
DESC – porządek malejący

Sortowanie rosnąco

* SELECT \* FROM nazwa\_tabeli ORDER BY nazwa\_kolumny;
* SELECT \* FROM nazwa\_tabeli ORDER BY nazwa\_kolumny ASC;

Sortowanie malejąco

* SELECT \* FROM nazwa\_tabeli ORDER BY nazwa\_kolumny DESC;

Sortowanie wielu kolumn

* SELECT \* FROM nazwa\_tabeli ORDER BY nazwa\_kolumny1 ASC, nazwa\_kolumny2 DESC;

**Operator BETWEEN**

Używany w celu uzyskania wartości z zadeklarowanego przedziału.

Osoby o id od 5 do 10

* SELECT \* FROM nazwa\_tabeli WHERE id BETWEEN 5 AND 10;

**Operator IN**

Jeżeli w wynikach mają zostać uwzględnione wartości z pewnego zbioru, a nie z przedziału można skorzystać z operatora IN

Osoby o id równym 5, 7 lub 9

* SELECT \* FROM nazwa\_tabeli WHERE id IN(5, 7, 9);

**Operator NOT IN**

Odwrotność operatora in

Osoby o id nie równym 5, 7 lub 9

* SELECT \* FROM nazwa\_tabeli WHERE id NOT IN (5, 7, 9);

**Klauzula TOP (LIMIT)**

Maksymalnie 5 rekordów

* SELECT \* FROM nazwa\_tabeli ORDER BY nazwa\_kolumny ASC LIMIT 5

**Warunek logiczny CHECK**

Atrybut CHECK pozwala na zdefiniowanie warunków ograniczających zakres danych wprowadzanych do określonej kolumny. Dla każdej kolumny można zdefiniować wiele warunków. Za pomocą operatorów AND, OR, NOT można konstruować złożone warunki.

Tworzenie tabeli, gdzie wiek musi być w zakresie od 0 do 150 i nazwisko nie może być Kowalski

* CREATE TABLE nazwa\_tabeli (id int AUTO\_INCREMENT NOT NULL PRIMARY KEY, nazwisko varchar(60), imie varchar(40), wiek int(2), CHECK (wiek>0 AND wiek<150), CHECK (nazwisko<>’Kowalski’));

Wprowadzanie klauzuli CHECK do istniejącej tabeli

* ALTER TABLE nazwa\_tabeli ADD CHECK (imie<>’Adam’);

Usuwanie atrybutu CHECK z tabeli

* ALTER TABLE nazwa\_tabeli DROP CONSTRAINT CONSTRAINT\_1;

Dodawanie ograniczenia z własną nazwą

* ALTER TABLE nazwa\_tabeli ADD CONSTRAINT nazwa\_ograniczenia CHECK (ograniczenie);

**Widoki (perspektywy)**

Tworzenie widoku tylko z kobietami

* CREATE VIEW nazwa\_widoku AS SELECT \* FROM nazwa\_tabeli WHERE imie LIKE ‘%a’;

Usuwanie widoku

* DROP VIEW nazwa\_widoku;

**Pobieranie danych z kilku tabel**

Pobieranie danych z tabel osoba i zamówienia (iloczyn kartezjański)

* SELECT osoba.id, osoba.imie, osoba.nazwisko, zamowienia.idklient, zamowienia.towar, zamowienia.ilosc, zamowienia.data FROM osoba, zamówienia ORDER BY imie;

Pełne złączenie tabel

* SELECT osoba.imie, osoba.nazwisko, zamowienia.idklient, zamowienia.towar, zamowienia.ilosc, zamowienia.data FROM osoba, zamówienia WHERE osoba.id=zamowienia.idklient ORDER BY nazwisko;

Aliasy zamiast pełnych nazw tabel

* SELECT o.id, o.imie, z.idklient, z.towar, z.ilosc, z.data FROM osoba AS o, zamówienia AS z WHERE o.id=z.idklient ORDER BY imie;

**Złączenia wewnętrzne**

* SELECT osoba.imie, osoba.nazwisko, zamówienia.idklient, zamówienia.idtowar, zamówienia.ilosc, zamówienia.data FROM osoba INNER JOIN zamówienia ON osoba.id=zamówienia.idklient

**Złączenia zewnętrzne**

Złączenia dzielimy na:

Zapytanie zwraca wszystkie wiersze z pierwszej tabeli i pasujące wiersze z drugiej tabeli

* LEFT OUTER JOIN

Zapytania zwraca wszystkie wiersze z drugiej tabeli i pasujące wiersze z pierwszej tabeli

* RIGHT OUTER JOIN

Zapytanie zwraca wszystkie wiersze pasujące i nie pasujące wiersze z obu tabel (SQL Server)

* FULL OUTER JOIN

Wyświetlanie rekordów dla których brakuje wpisów w tabeli zamówienia

* SELECT o.imie, o.nazwisko, z.data, z.ilosc FROM osoba AS o LEFT JOIN zamówienia AS z ON o.id=z.idklient WHERE z.id IS NULL

**Złączenie własne**

Można wykorzystać do stwierdzenia czy dany klient został wpisany do bazy wielokrotnie.

* SELECT kl1.id, kl2.id, kl1.imie, kl2.imie, kl1.nazwisko, kl2.nazwisko FROM klient AS kl1 CROSS JOIN klient AS kl2;
* SELECT kl1.id, kl2.id, kl1.imie, kl2.imie, kl1.nazwisko, kl2.nazwisko FROM klient AS kl1 CROSS JOIN klient AS kl2 WHERE kl1.nazwisko=kl2.nazwisko AND kl1.imie=kl2.imie AND kl1.id<>kl2.id

**Przeprowadzanie obliczeń matematycznych**

Zapytanie zwraca wynik działania

* SELECT 3 \* 10;

Zapytanie zwraca akutalną datę

* SELECT Curdate();

**Grupowanie danych**

Z bazy danych możemy uzyskać informacje na temat średnich wartości wprowadzonych zamówień, liczby pracowników, liczby towarów, spełniających określone kryteria.

Uzyskanie takich informacji umożliwia zestaw dostępnych w SQL funkcji agregujących i statystycznych.

Funkcje zwracają pojedyncze wartości i są wywoływane w instrukcji SELECT.

Funkcje agregujące są jednymi z najważniejszych narzędzi relacyjnych baz danych.

Oblicza wartość średnią z wartości zwróconych przez zapytanie

* SELECT AVG(cena) AS ‘Srednia cena zakupu’ FROM zamówienia

Zwraca liczbę wierszy w kolumnie

* SELECT COUNT(nazwa\_kolumny lub \*) FROM nazwa\_tabeli

Zwraca sumę wartości w grupie dla wskazanej kolumny

* SELECT SUM(nazwa\_kolumny) FROM nazwa\_tabeli

Zwraca największą wartość w grupie dla wskazanej kolumny

* SELECT MAX(nazwa\_kolumny) FROM nazwa\_tabeli

Zwraca najmniejsza wartość w grupie dla wskazanej kolumny

* SELECT MIN(nazwa\_kolumny) FROM nazwa\_tabeli

Zwraca ilośc Adamów w tabeli

* SELECT COUNT(\*) AS ‘Liczba pracownikow’ FROM pracownik WHERE imie=’Adam’;

Użycie kilku funkcji w 1 zapytaniu

* SELECT MIN(cena) AS ‘Najtansza cena’, MAX(cena) AS ‘Najdrozsza cena’ FROM zamówienia

Jeżeli funkcja agregujaca w swoich obliczeniach uwzględnia tylko wartości niepowtarzające się to argumentem funkcji staje się słowo kluczowe DISTINCT

* SELECT COUNT(DISTINCT nazwisko) AS ‘Wprowadzone nazwiska’ FROM pracownik WHERE telefon IS NOT NULL;

**Tworzenie kopii zapasowych**

Przed wykonaniem kopii należy zablokować bazę danych poleceniem

* FLUSH TABLES

Wykonanie pełnej kopii baz danych o nazwie firma

* mysqldump –u root –p > plik.sql

Wykonanie pełnej kopii baz danych o nazwie firma w określonej lokalizacji

* mysqldump –u root –p firma > c:/backup/plik.sql

Wykonanie pełnej kopii wszystkich baz danych znajdujących się na serwerze

* mysqldump –u root –p –all-databases > c:/backup/plik.sql

Składnia polecenia do przywracania bazy danych

* mysql –u root –p firma < plik.sql

**Funkcje tekstowe**

* LOWER() – konwertuje na małe litery
* UPPER() – konwertuje na duże litery
  + SELECT imie, LOWER(imie) FROM osoba;
* LEFT() – zwraca znaki z lewej strony tekstu
  + SELECT imie, LEFT(imie, 3) FROM osoba;
* RIGHT() – zwraca znaki z prawej strony tekstu
  + SELECT imie, RIGHT(imie, 3) FROM osoba;
* LENGTH() – zwraca długość tekstu
  + SELECT imie, LENGTH(imie) FROM osoba;
* LTRIM() – usuwa spacje z lewej strony tekstu
* RTRIM() – usuwa spacje z prawej strony tekstu
  + SELECT LTRIM(‘ Ala ma Kota ‘), RTRIM(‘ Ala ma Kota ‘);
* SUBSTRING() – pobiera fragment łańcucha znaków (od, do)
  + SELECT imie, SUBSTRING(imie, 2, 5) FROM osoba;

**Funkcje liczbowe**

* ROUND() – polega na zaokrągleniu liczby zgodnie z wartością podaną po przecinku
  + SELECT ROUND(AVG(cena), 2) FROM towar;
* TRUNCATE() – powoduje obcięcie liczby do określonej liczby cyfr po przecinku. Pierwszy parametr jest liczbą do obcięcia, drugi wskazuje, do ilu pozycji chcemy liczbę skrócić
  + SELECT TRUNCATE(AVG(cena), 2) FROM towar;
* SQRT() – zwraca pierwiastek kwadratowy podanej liczby
  + SELECT SQRT(225);
* PI(), SIN(), COS(), TAN() – funkcje matematyczne
* RAND() – pseudolosowa liczba od 0 do 1

**Funkcje daty i czasu**

* CURDATE() – bieżąca data
* CURTIME() – bieżący czas
  + SELECT CURDATE(), CURTIME()
* NOW() – zwraca aktualną datę i czas systemowy
  + SELECT NOW();
* DAYOFMONTH(), DAYOFWEEK(), DAYOFYEAR() – dzień miesiąca, tygodnia (1 – niedziela, 7 – sobota), roku
* DAY(), MONTH(), YEAR() – zwracają dzień, miesiąc i rok z daty przekazanej jako parametr wywołania
  + SELECT DAY(‘2022-03-28’)
* DAYNAME(), MONTHNAME() – zwracają nazwę dnia tygodnia i miesiąca daty będącej argumentem wywołania

**Definiowanie klucza obcego**

Aby uniknąć problemu z niespójnymi danymi (Klient o id równym 2 ma przypisany adres id równym 13, gdy w tabeli z adresami nie ma rekordu o tym identyfikatorze), należałoby za każdym razem sprawdzać czy w tabeli adresy są odpowiednie rekordy.

Taka czynność może być wykonywana automatycznie przez serwer bazy danych, pod warunkiem, że odpowiednio są zdefiniowane więzy integralności (warunki które muszą spełniać dane w bazie danych np. umożliwienie wprowadzenie odkreślonych wartości do kolumny).

Ogólna konstrukcja wprowadzająca ograniczenia ze względu na klucze obce

* CONSTRAINT nazwa FOREIGN KEY (kolumna,…) REFERENCES nazwa\_tabeli(kolumna,…)

Tworzenie klucza obcego wraz z tabelą

* CREATE TABLE zakupyobcy (id int(11) AUTO\_INCREMENT PRIMARY KEY, ilość int(5), data date, idklient int(11), CONSTRAINT kluczobcy FOREIGN KEY(idklient) REFERENCES klient(id))

Dodawanie klucza obcego do istniejącej tabeli

* ALTER TABLE zakup ADD CONSTRAINT klucz\_o FOREIGN KEY (idklient) REFERENCES klient(id)

Usuwanie klucza obcego

* ALTER TABLE zakupy DROP FOREIGN KEY klucz\_obcy