

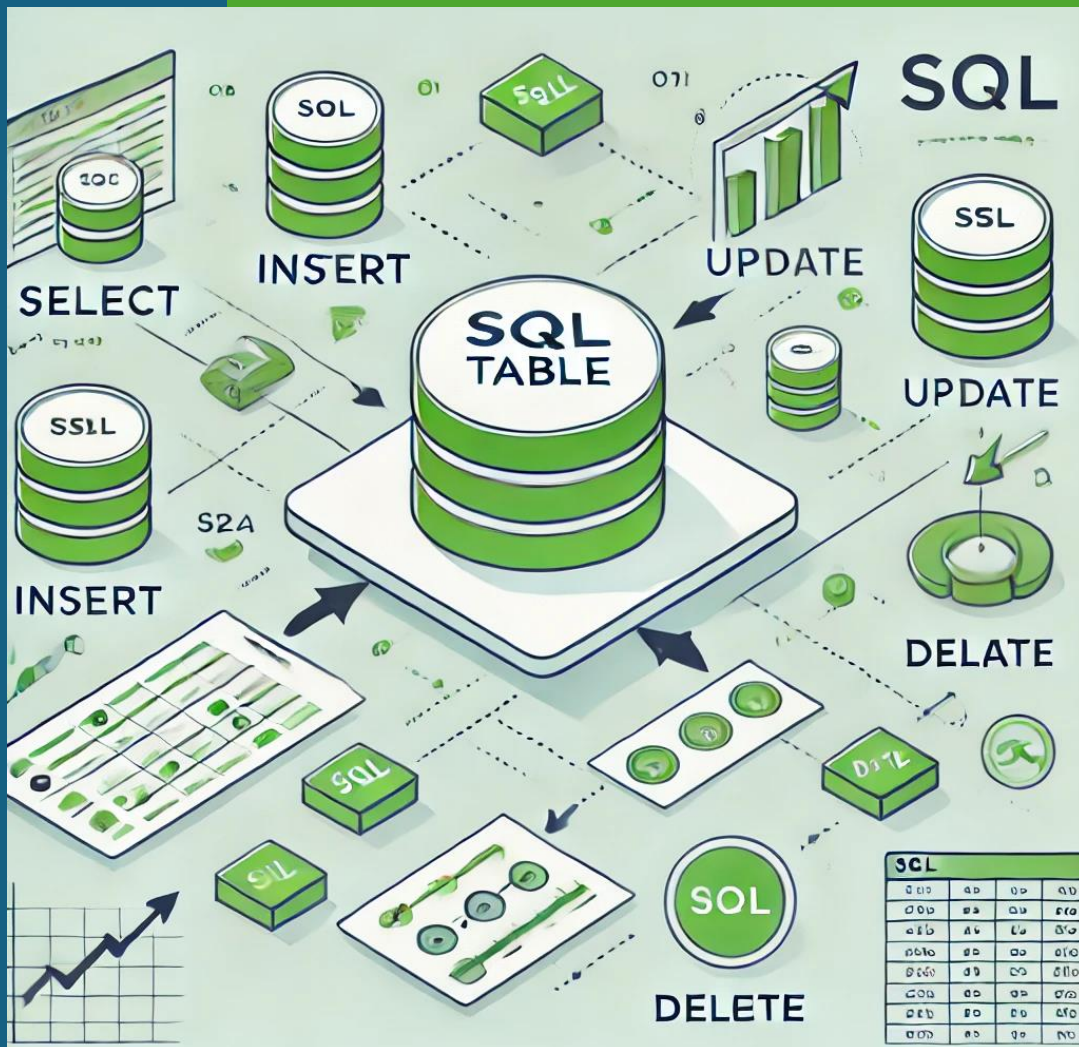
Temat 4 .

Język SQL

dr inż. Michał Malinowski

<https://www.drmalinowski.edu.pl>

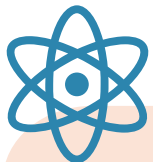
PODSTAWY BAZ DANYCH



Zagadnienia:

- Język SQL
- DQL – zapytania o dane
- DDL – definicja struktury
- DML – edycja danych
- DCL – uprawnienia i dostęp
- TCL – zarządzanie transakcjami
- Ćwiczenie praktyczne !

Komputery kwantowe



Komputery kwantowe

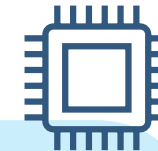
Operują na kubitach, które mogą być w superpozycji 0 i 1

Wykorzystują zjawiska kwantowe takie jak superpozycja i splątanie, co umożliwia równoległe przetwarzanie informacji.

Przeznaczone do specjalistycznych obliczeń, takich jak kryptografia, sztuczna inteligencja, symulacje kwantowe.

Muszą być utrzymywane w bardzo niskich temperaturach bliskich zeru absolutnemu, co ogranicza ich praktyczne zastosowanie.

Są niestabilne i podatne na zakłócenia z otoczenia, co utrudnia ich utrzymanie i wykorzystanie.



Komputery tradycyjne

Operują na bitach, które mogą przyjmować wartość 0 lub 1.

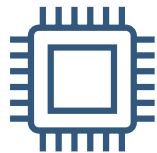
Przetwarzają dane sekwencyjnie, wykonując operacje krok po kroku.

Są powszechnie używane w codziennych zadaniach, takich jak przetwarzanie tekstu, przeglądanie internetu i gry komputerowe.

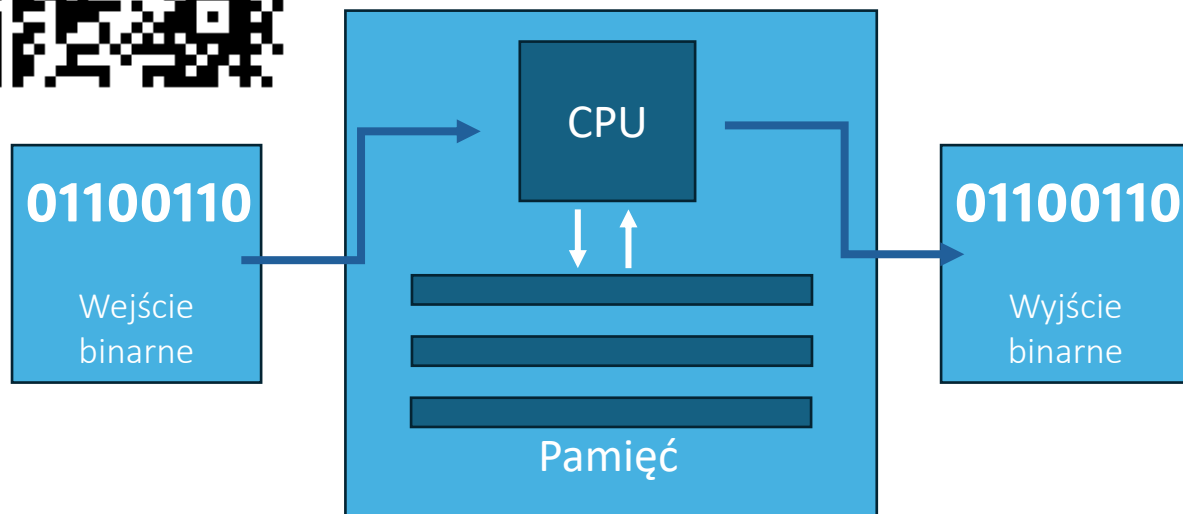
Osiągają swoje limity fizyczne przy zmniejszaniu rozmiaru tranzystorów, co prowadzi do wielu problemów.

Są stabilne i niezawodne, z łatwo dostępnymi częściami i technologią.

Komputer neuromorficzny



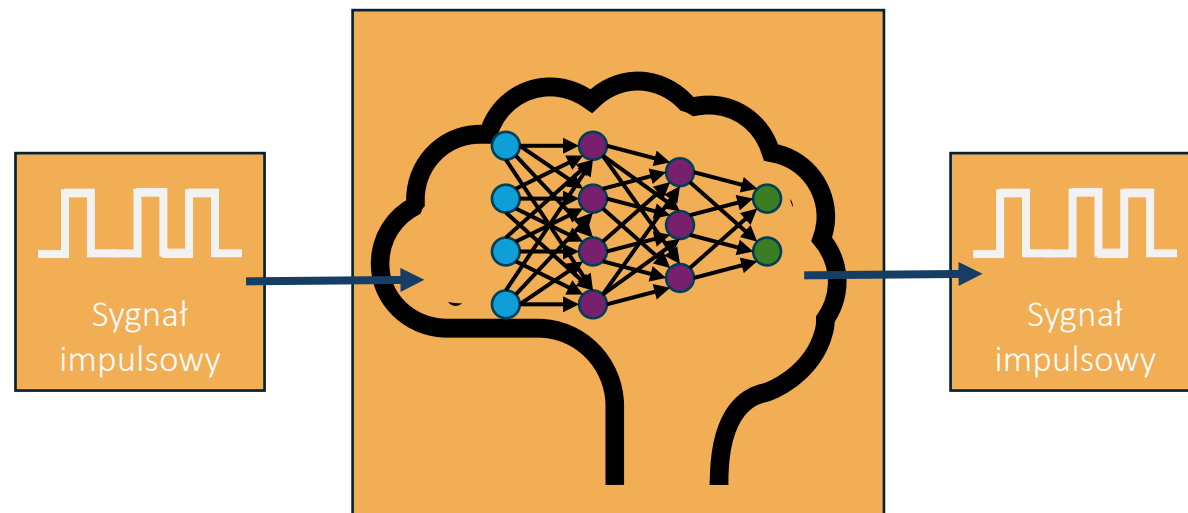
Architektura von Neumanna



Przetwarzanie sekwencyjne
Oddzielone przetwarzanie i pamięć
Kod w postaci instrukcji binarnych
Dane binarne
Skoordynowanie za pomocą zegara
(sterowane zegarem)



Architektura neuromorficzna



Przetwarzanie równoległe na dużą skalę
Połączone przetwarzanie i pamięć
Sieć neuronowa oparta na impulsach
Sygnały impulsowe
Asynchroniczność
(sterowanie zdarzeniami)

Język SQL

SQL (Structured Query Language)

to strukturalny język zapytań, który jest standardem w zarządzaniu i manipulacji danymi w relacyjnych bazach danych. Umożliwia tworzenie, modyfikowanie, odczytywanie oraz zarządzanie danymi przechowywanymi w tabelach, które są połączone relacjami.

Cechy

Deklaratywność:

- Jest językiem deklaratywnym, co oznacza, że użytkownik określa, co chce osiągnąć, ale nie musi wskazywać, jak to zrobić.

Spójność i integralność danych:

- Pozwala na zachowanie integralności danych, zapobiegając duplikatom oraz błędom związanym z relacjami między danymi.

Obsługa transakcji:

- Umożliwia zarządzanie transakcjami poprzez mechanizmy ACID (Atomicity, Consistency, Isolation, Durability)

Obsługa wielu baz danych:

- Jest standardem wspieranym przez większość relacyjnych systemów baz danych

Język SQL

Zalety

Łatwość użycia:

SQL jest stosunkowo prostym do nauczenia językiem, z jasną składnią i dobrze zdefiniowanymi funkcjami.

Wieloplatformowość:

Standard SQL jest wspierany przez wiele systemów zarządzania bazami danych, co ułatwia migrację i integrację danych.

Potężne narzędzie analityczne:

SQL umożliwia tworzenie złożonych zapytań, agregacji danych, tworzenie podzapytań oraz analizowanie danych w sposób efektywny.

Wady

Słabość w obsłudze danych nieustrukturyzowanych:

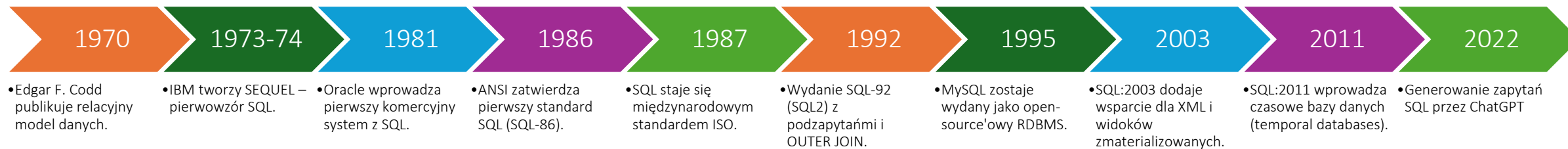
Relacyjne bazy danych, mają ograniczoną wydajność przy danych nieustrukturyzowanych, takich jak multimedia, duże pliki tekstowe.

Ograniczenia skalowalności:

Relacyjne bazy danych mają problemy z wydajnością przy pracy z ogromnymi ilościami danych (Big Data), gdzie bardziej wydajne są bazy NoSQL.

Język SQL

Historia



Język SQL

Grupy komend

DQL (Data Query Language)	DDL (Data Definition Language)	DML (Data Manipulation Language)	DCL (Data Control Language)	TCL (Transaction Control Language)
Pobieranie danych z bazy danych.	Definiowanie i modyfikowanie struktury bazy danych.	Manipulowanie danymi w tabelach.	Zarządzanie uprawnieniami dostępu do danych.	Zarządzanie transakcjami w bazie danych.
SELECT, WHERE, INNER	CREATE, ALTER, DROP	INSERT, UPDATE, DELETE	GRANT, REVOKE	COMMIT, ROLLBACK, SAVEPOINT

DQL – zapytania o dane

DQL (Data Query Language)

jest jedną z głównych grup komend SQL, której celem jest pobieranie danych z bazy danych.

DQL umożliwia użytkownikom formułowanie zapytań do bazy, aby odczytać informacje z tabel zgodnie z określonymi warunkami.

DQL – zapytania o dane

- **SELECT [kolumny]:**
Wybiera kolumny, które mają zostać zwrócone w wynikach zapytania.
- **FROM [tabela lub tabele]:**
Określa tabelę lub tabele, z których mają być pobrane dane.
- **JOIN [inna tabela] ON [warunek]:**
Łączy dane z dwóch lub więcej tabel na podstawie określonego warunku.
- **WHERE [warunki filtrowania]:**
Filtrowanie rekordów na podstawie określonych warunków logicznych.
- **GROUP BY [kolumna]:**
Grupuje wyniki na podstawie jednej lub więcej kolumn.
- **HAVING [warunek na grupę]:**
Filtrowanie grup po wykonaniu operacji grupowania.
- **ORDER BY [kolumna] [ASC/DESC]:**
Sortowanie wyników w porządku rosnącym (ASC) lub malejącym (DESC).

```
SELECT [kolumny]
FROM [tabela lub tabele]
JOIN [inna tabela] ON [warunek]
WHERE [warunki filtrowania]
GROUP BY [kolumna]
HAVING [warunek na grupę]
ORDER BY [kolumna] [ASC/DESC];
```

DQL – zapytania o dane

Tabela 1: Klienci

KlientID	Imie	Nazwisko	Miasto
1	Jan	Nowak	Warszawa
2	Anna	Kowalska	Kraków
3	Piotr	Wiśniewski	Gdańsk
4	Ewa	Kwiatkowska	Warszawa
5	Marek	Kowalski	Wrocław

```
SELECT Imie, Nazwisko
FROM Klienci;
```

Imie	Nazwisko
Jan	Nowak
Anna	Kowalska
Piotr	Wiśniewski
Ewa	Kwiatkowska
Marek	Kowalski

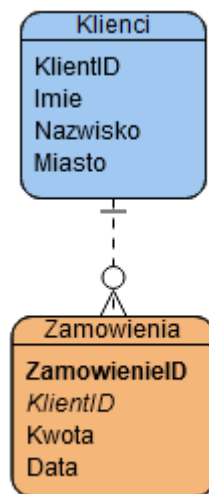


Tabela 2: Zamowienia

ZamowienieID	KlientID	Kwota	Data
101	1	1500	10.01.2023
102	2	2500	15.02.2023
103	1	2000	05.03.2023
104	3	3000	01.04.2023
105	4	1200	12.05.2023
106	4	5000	01.06.2023

```
SELECT Klienci.Imie, Klienci.Nazwisko, Zamowienia.Kwota
FROM Klienci
INNER JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID;
```

Imie	Nazwisko	Kwota
Jan	Nowak	1500
Jan	Nowak	2000
Anna	Kowalska	2500
Piotr	Wiśniewski	3000
Ewa	Kwiatkowska	1200
Ewa	Kwiatkowska	5000

DQL – zapytania o dane

Imie	Nazwisko	Kwota
Jan	Nowak	1500
Jan	Nowak	2000
Anna	Kowalska	2500
Piotr	Wiśniewski	3000
Ewa	Kwiatkowska	1200
Ewa	Kwiatkowska	5000

KlientID	Imie	Nazwisko	Miasto
1	Jan	Nowak	Warszawa
2	Anna	Kowalska	Kraków
3	Piotr	Wiśniewski	Gdańsk
4	Ewa	Kwiatkowska	Warszawa
5	Marek	Kowalski	Wrocław

```
SELECT Klienci.Imie, Klienci.Nazwisko, Zamowienia.Kwota
FROM Klienci
INNER JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID
WHERE Klienci.Miasto = 'Warszawa';
```

Imie	Nazwisko	Kwota
Jan	Nowak	1500
Jan	Nowak	2000
Ewa	Kwiatkowska	1200
Ewa	Kwiatkowska	5000

```
SELECT Klienci.Imie, Klienci.Nazwisko, SUM(Zamowienia.Kwota) AS SumaZamowien
FROM Klienci
INNER JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID
WHERE Klienci.Miasto = 'Warszawa'
GROUP BY Klienci.Imie, Klienci.Nazwisko;
```

Imie	Nazwisko	SumaZamowien
Jan	Nowak	3500
Ewa	Kwiatkowska	6200

DQL – zapytania o dane

Imie	Nazwisko	SumaZamowien
Jan	Nowak	3500
Ewa	Kwiatkowska	6200

```
SELECT Klienci.Imie, Klienci.Nazwisko, SUM(Zamowienia.Kwota) AS SumaZamowien
FROM Klienci
INNER JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID
WHERE Klienci.Miasto = 'Warszawa'
GROUP BY Klienci.Imie, Klienci.Nazwisko
HAVING SUM(Zamowienia.Kwota) > 1000;
```

Imie	Nazwisko	SumaZamowien
Jan	Nowak	3500
Ewa	Kwiatkowska	6200

```
SELECT Klienci.Imie, Klienci.Nazwisko, SUM(Zamowienia.Kwota) AS SumaZamowien
FROM Klienci
INNER JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID
WHERE Klienci.Miasto = 'Warszawa'
GROUP BY Klienci.Imie, Klienci.Nazwisko
HAVING SUM(Zamowienia.Kwota) > 1000
ORDER BY SumaZamowien DESC;
```

A
↓
Z

Imie	Nazwisko	SumaZamowien
Ewa	Kwiatkowska	6200
Jan	Nowak	3500

DQL – zapytania o dane

INNER JOIN

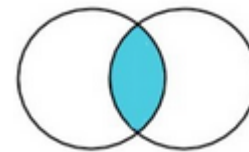
Zwraca tylko te rekordy, które mają dopasowane wartości w obu tabelach. Jeśli nie ma dopasowania, rekordy nie zostaną uwzględnione w wyniku.

Zastosowanie: Gdy chcemy uzyskać tylko te dane, które są obecne w obu tabelach.

Szybsze wyniki ale mniej !

```
SELECT Klienci.Imie, Zamowienia.Data  
FROM Klienci  
INNER JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID;
```

Imie	Data
Jan	10.01.2023
Jan	05.03.2023
Anna	15.02.2023
Piotr	01.04.2023



DQL – zapytania o dane

LEFT JOIN

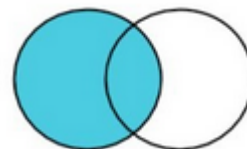
Zwraca wszystkie rekordy z lewej tabeli oraz dopasowane rekordy z prawej tabeli. Jeśli w prawej tabeli brak dopasowania, zwraca NULL w miejscach brakujących danych.

Zastosowanie: Gdy chcemy uzyskać wszystkie dane z lewej tabeli, nawet jeśli nie mają odpowiadających rekordów w prawej tabeli.

Wolniejsze wyniki ale więcej !

```
SELECT Klienci.Imie, Zamowienia.Data  
FROM Klienci  
LEFT JOIN Zamowienia ON Klienci.KlientID = Zamowienia.KlientID;
```

Imie	Data
Jan	10.01.2023
Jan	05.03.2023
Anna	15.02.2023
Piotr	01.04.2023
Marek	NULL



DQL – zapytania o dane

Klauzula WHERE

Operator	Opis	Przykład
=, >, <, <>, >=, <=	Porównanie wartości.	<i>WHERE wiek >= 18</i>
AND / OR	Łączenie wielu warunków.	<i>WHERE wiek > 30 AND miasto = 'Warszawa'</i>
NOT	Odwraca wynik warunku.	<i>WHERE NOT miasto = 'Warszawa'</i>
IN	Sprawdza, czy wartość znajduje się na liście wartości.	<i>WHERE miasto IN ('Warszawa', 'Kraków')</i>
BETWEEN	Sprawdza, czy wartość mieści się w określonym zakresie.	<i>WHERE wiek BETWEEN 18 AND 65</i>
LIKE	Wyszukiwanie według wzorca.	<i>WHERE nazwisko LIKE 'Ko%'</i>
IS NULL / IS NOT NULL	Sprawdza, czy wartość jest NULL lub NOT NULL.	<i>WHERE telefon IS NULL</i>
Podzapytania	Zagnieżdżone zapytanie, które zwraca wartości do porównania.	<i>WHERE KlientID IN (SELECT KlientID FROM Zamowienia)</i>
EXISTS	Sprawdza, czy podzapytanie zwraca jakiegolwiek wiersze.	<i>WHERE EXISTS (SELECT * FROM Zamowienia)</i>

DQL – zapytania o dane

ORDER BY, LIMIT i DISTINCT w SQL:

ORDER BY służy do sortowania wyników zapytania w porządku rosnącym (**ASC**) lub malejącym (**DESC**)

```
SELECT kolumny
FROM tabela
ORDER BY kolumna1 [ASC|DESC], kolumna2 [ASC|DESC];
```

LIMIT służy do ograniczenia liczby zwróconych wierszy w wyniku zapytania

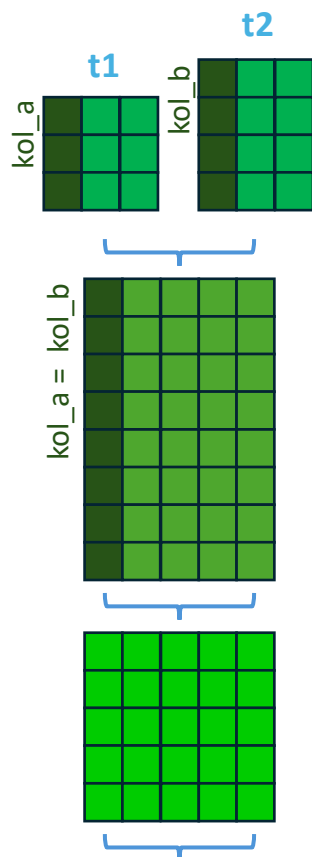
```
SELECT kolumny
FROM tabela
LIMIT liczba_wierszy;
```

DISTINCT służy do eliminowania duplikatów i zwracania unikalnych wartości z określonych kolumn.

```
SELECT DISTINCT kolumna
FROM tabela;
```

```
SELECT DISTINCT miasto
FROM Klienci
ORDER BY miasto ASC
LIMIT 3;
```

SQL - SELECT



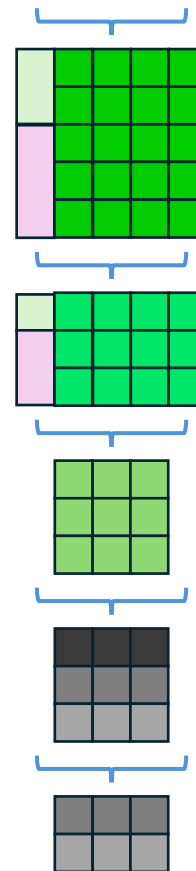
SELECT kol_a, kol_b

FROM t1

JOIN t2

ON t1.kol_a = t2.kol_b

WHERE warunki



GROUP BY kol

HAVING warunki

ORDER BY kol ASC/DESC

LIMIT liczba

DQL – zapytania o dane

Funkcje agregujące

Funkcja	Opis	Przykład
COUNT()	Zlicza liczbę wierszy (lub wartości nie-NULL w danej kolumnie).	<i>SELECT COUNT(*) FROM Klienci;</i>
SUM()	Sumuje wartości w danej kolumnie (dla wartości numerycznych).	<i>SELECT SUM(kwota) FROM Zamowienia;</i>
AVG()	Oblicza średnią wartość w danej kolumnie (dla wartości numerycznych).	<i>SELECT AVG(wiek) FROM Klienci;</i>
MIN()	Zwraca najmniejszą wartość w danej kolumnie.	<i>SELECT MIN(kwota) FROM Zamowienia;</i>
MAX()	Zwraca największą wartość w danej kolumnie.	<i>SELECT MAX(wiek) FROM Klienci;</i>

DDL – definicja struktury

DDL (Data Definition Language)

to podzbiór SQL, który służy do definiowania, modyfikowania i usuwania struktur danych w bazie danych, takich jak tabele, widoki, indeksy oraz inne obiekty.

Komendy DDL są odpowiedzialne za tworzenie i zarządzanie schematami baz danych.

DDL – definicja struktury

Typy danych w MySQL

Typ danych	Opis	Zakres	Przykład
INT	Liczba całkowita.	-2,147,483,648 do 2,147,483,647 (SIGNED) lub 0 do 4,294,967,295 (UNSIGNED)	INT(11) - 42
TINYINT	Bardzo mała liczba całkowita.	-128 do 127 (SIGNED) lub 0 do 255 (UNSIGNED)	TINYINT(1) - 1
SMALLINT	Mała liczba całkowita.	-32,768 do 32,767 (SIGNED) lub 0 do 65,535 (UNSIGNED)	SMALLINT(5) - 32000
BIGINT	Duża liczba całkowita.	-9,223,372,036,854,775,808 do 9,223,372,036,854,775,807 (SIGNED)	BIGINT(20) - 9000000000
FLOAT	Liczba zmiennoprzecinkowa o pojedynczej precyzji.	-3.402823466E+38 do -1.175494351E-38 oraz 1.175494351E-38 do 3.402823466E+38	FLOAT(7,3) - 123.456
DOUBLE	Liczba zmiennoprzecinkowa o podwójnej precyzji.	-1.7976931348623157E+308 do -2.2250738585072014E-308 oraz 2.2250738585072014E-308 do 1.7976931348623157E+308	DOUBLE(16,4) - 12345.6789
DECIMAL	Liczba dziesiętna o określonej precyzji.	Zakres zależny od definicji, np. DECIMAL(5,2) mieści wartości od -999.99 do 999.99	DECIMAL(10,2) - 98765.43
CHAR	Stały ciąg znaków o określonej długości.	Do 255 znaków	CHAR(5) - 'Hello'
VARCHAR	Zmienny ciąg znaków o określonej długości.	Do 65,535 znaków	VARCHAR(255) - 'Przykład'
TEXT	Tekst o zmiennej długości.	Do 65,535 znaków	TEXT - 'Lorem ipsum...'
DATE	Data (rok, miesiąc, dzień).	'1000-01-01' do '9999-12-31'	DATE - '2023-09-19'
DATETIME	Data i czas (rok, miesiąc, dzień, godzina, minuta, sekunda).	'1000-01-01 00:00:00' do '9999-12-31 23:59:59'	DATETIME - '2023-09-19 10:30:00'
TIMESTAMP	Znacznik czasu Unix (sekundy od 1970-01-01).	'1970-01-01 00:00:01' do '2038-01-19 03:14:07'	TIMESTAMP - '2023-09-19 10:30:00'
TIME	Czas (godzina, minuta, sekunda).	'-838:59:59' do '838:59:59'	TIME - '12:30:45'
YEAR	Rok.	'1901' do '2155'	YEAR - '2023'
ENUM	Zmienna przyjmująca jedną z określonych wartości.	Zdefiniowane przez użytkownika wartości.	ENUM('mały', 'średni', 'duży')
SET	Zbiór wartości zdefiniowanych przez użytkownika.	Zdefiniowane przez użytkownika, maks. 64 wartości.	SET('HTML', 'CSS', 'SQL')

DDL – definicja struktury

CREATE

Tworzenie nowych obiektów w bazie danych (tabele, widoki, indeksy, bazy danych).

Tworzenie tabeli

```
CREATE TABLE nazwa_tabeli (  
    kolumna1 typ_danych [opcje],  
    kolumna2 typ_danych [opcje],  
    ...  
    kolumnaN typ_danych [opcje]  
);
```

Tworzenie widoku

```
CREATE VIEW nazwa_widoku AS  
SELECT kolumna1, kolumna2, ...  
FROM nazwa_tabeli  
WHERE warunek;
```

Tworzenie indeksu

```
CREATE INDEX nazwa_indeksu  
ON nazwa_tabeli (kolumna1, kolumna2, ...);
```

```
CREATE TABLE Pracownicy (  
    ID INT PRIMARY KEY,  
    Imie VARCHAR(50) NOT NULL,  
    Stanowisko VARCHAR(50),  
    Wiek INT  
);
```

```
CREATE VIEW KlienciWarszawa AS  
SELECT Imie, Nazwisko, Miasto  
FROM Klienci  
WHERE Miasto = 'Warszawa';
```

```
CREATE INDEX idx_MiastoNazwisko  
ON Klienci (Miasto, Nazwisko);
```

DDL – definicja struktury

ALTER

Zmiana struktury istniejących obiektów w bazie danych (np. dodanie kolumny do tabeli, zmiana typu danych).

Dodawanie kolumny

```
ALTER TABLE nazwa_tabeli  
ADD kolumna typ_danych [opcje];
```

```
ALTER TABLE Pracownicy  
ADD Adres VARCHAR(100);
```

Usuwanie kolumny

```
ALTER TABLE nazwa_tabeli  
DROP COLUMN kolumna;
```

```
ALTER TABLE Pracownicy  
DROP COLUMN Adres;
```

Modyfikacji kolumny
(zmiana typu danych)

```
ALTER TABLE nazwa_tabeli  
MODIFY kolumna typ_danych [opcje];
```

```
ALTER TABLE Pracownicy  
MODIFY Wiek VARCHAR(3);
```

DDL – definicja struktury

DROP

Usuwanie obiektów z bazy danych (tabele, widoki, bazy danych). Usunięcie jest trwałe i nieodwracalne.

```
DROP TABLE Pracownicy;
```

TRUNCATE

Usunięcie wszystkich danych z tabeli, ale bez usuwania samej struktury tabeli.

```
TRUNCATE TABLE Pracownicy;
```

DROP

Usuwa całą tabele oraz jej strukturę.

TRUNCATE

Usuwa wszystkie dane z tabeli, ale zachowuje strukturę tabeli.

DML – edycja danych

DML (Data Manipulation Language)

to grupa komend SQL, które służą do manipulacji danymi przechowywanymi w bazie danych.

Komendy DML umożliwiają dodawanie, modyfikowanie oraz usuwanie rekordów w tabelach.

DDL – edycja danych

INSERT

dodawanie nowych rekordów do tabeli.

```
INSERT INTO tabela (kolumna1, kolumna2, ...)
VALUES (wartość1, wartość2, ...);
```

```
INSERT INTO Pracownicy (ID, Imie, Stanowisko)
VALUES (1, 'Jan', 'Manager');
```

UPDATE

aktualizacja istniejących danych.

```
UPDATE tabela
SET kolumna1 = wartość1, kolumna2 = wartość2, ...
WHERE warunek;
```

```
UPDATE Pracownicy
SET Stanowisko = 'Kierownik'
WHERE ID = 1;
```

DELETE

usuwanie rekordów z tabeli.

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

```
DELETE FROM Pracownicy
WHERE ID = 1;
```

DCL – uprawnienia i dostęp

DCL (Data Control Language)

to grupa komend SQL służąca do zarządzania uprawnieniami i kontrolą dostępu do danych w bazach danych.

Komendy DCL umożliwiają administratorom baz danych nadawanie i odbieranie uprawnień użytkownikom oraz grupom użytkowników, co pozwala na kontrolowanie, kto i jakie operacje może wykonywać na danych.

DCL – uprawnienia i dostęp

Grupy uprawnień

Grupa uprawnień	Opis	Przykładowe komendy
Uprawnienia do danych	Uprawnienia dotyczące operacji na danych przechowywanych w tabelach.	SELECT, INSERT, UPDATE, DELETE
Uprawnienia do struktury	Uprawnienia związane z tworzeniem i modyfikowaniem struktury baz danych.	CREATE, ALTER, DROP, TRUNCATE
Uprawnienia administracyjne	Uprawnienia pozwalające na zarządzanie użytkownikami oraz ich uprawnieniami.	GRANT, REVOKE
Uprawnienia do procedur	Uprawnienia dotyczące korzystania z procedur składowanych, funkcji, wyzwalaczy.	EXECUTE
Uprawnienia do widoków	Uprawnienia związane z korzystaniem z widoków (wirtualnych tabel).	SELECT, INSERT, UPDATE, DELETE na widokach
Uprawnienia do indeksów	Uprawnienia dotyczące tworzenia i zarządzania indeksami.	CREATE INDEX, DROP INDEX
Uprawnienia do transakcji	Uprawnienia związane z zarządzaniem transakcjami w bazie danych.	COMMIT, ROLLBACK

DCL – uprawnienia i dostęp

GRANT

służy do przyznawania uprawnień użytkownikom do wykonywania określonych operacji, takich jak odczyt, zapis, modyfikacja danych lub zarządzanie strukturą bazy.

Uprawnienia mogą dotyczyć określonych tabel, widoków, procedur składowanych, lub innych obiektów w bazie danych.

```
GRANT <uprawnienia> ON <obiekt> TO <użytkownik> [WITH GRANT OPTION];
```

<uprawnienia>: Rodzaj uprawnień, np. SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES.

<obiekt>: Nazwa tabeli, widoku, procedury itp.

<użytkownik>: Nazwa użytkownika lub roli.

WITH GRANT OPTION: Pozwala użytkownikowi przekazywać przyznane uprawnienia innym użytkownikom.

```
GRANT SELECT ON Pracownicy TO Użytkownik;
```

```
GRANT SELECT, INSERT ON Zamówienia TO Jan;
```

```
GRANT ALL PRIVILEGES ON DATABASE MojaBaza TO Admin;
```

```
GRANT SELECT ON Klienci TO Użytkownik WITH GRANT OPTION;
```

DCL – uprawnienia i dostęp

MySQL

Globalne uprawnienia (Zaznacz wszystkie / Odznacz wszystkie)

Uwaga: Uprawnienia MySQL są oznaczone w języku angielskim

Dane	Struktura	Administracja	Ograniczenia zasobów
☐ SELECT	☐ CREATE	☐ GRANT	<i>Uwaga: Ustawienie tych opcji na 0 (zero) usuwa ograniczenie.</i>
☐ INSERT	☐ ALTER	☐ SUPER	MAX QUERIES PER HOUR 0
☐ UPDATE	☐ INDEX	☐ PROCESS	MAX UPDATES PER HOUR 0
☐ DELETE	☐ DROP	☐ RELOAD	MAX CONNECTIONS PER HOUR 0
☐ FILE	☐ CREATE TEMPORARY TABLES	☐ SHUTDOWN	MAX USER_CONNECTIONS 0
	☐ SHOW VIEW	☐ SHOW DATABASES	
	☐ CREATE ROUTINE	☐ LOCK TABLES	
	☐ ALTER ROUTINE	☐ REFERENCES	
	☐ EXECUTE	☐ REPLICATION CLIENT	
	☐ CREATE VIEW	☐ REPLICATION SLAVE	
	☐ EVENT	☐ CREATE USER	
	☐ TRIGGER		

DCL – uprawnienia i dostęp

REVOKE

pozwała na cofnięcie wcześniej przyznanych uprawnień użytkownikowi.

Odbiera prawa do wykonywania operacji na danych lub obiektach w bazie danych.

```
REVOKE <uprawnienia> ON <obiekt> FROM <użytkownik>;
```

<uprawnienia>: Rodzaj uprawnień, które chcemy cofnąć.

<obiekt>: Nazwa tabeli, widoku, procedury itp.

<użytkownik>: Nazwa użytkownika lub roli

```
REVOKE INSERT ON Pracownicy FROM Użytkownik;
```

```
REVOKE ALL PRIVILEGES ON DATABASE MojaBaza FROM Jan;
```

```
REVOKE <uprawnienia> ON <obiekt> FROM <użytkownik>;
```

TCL – zarządzanie transakcjami

TCL (Transaction Control Language)

służy do zarządzania transakcjami w bazie danych, co umożliwia kontrolowanie grupy operacji wykonywanych na danych w sposób gwarantujący spójność i integralność bazy.

Transakcje pozwalają wykonać zestaw operacji na bazie danych w sposób atomowy – wszystkie operacje muszą zostać zakończone pomyślnie, aby zmiany zostały wprowadzone, lub żadne nie zostaną wykonane, jeśli wystąpi błąd.

TCL – zarządzanie transakcjami

BEGIN TRANSACTION

to komenda rozpoczynająca nową transakcję w bazie danych.

Wszystkie operacje wykonywane po komendzie **BEGIN** będą traktowane jako część tej transakcji, aż do momentu zatwierdzenia jej przez **COMMIT** lub cofnięcia przez **ROLLBACK**.

```
BEGIN TRANSACTION;  
-- Operacje w ramach transakcji  
-- INSERT, UPDATE, DELETE, etc.  
COMMIT; -- lub ROLLBACK w przypadku błędu
```

COMMIT

zatwierdza wszystkie zmiany dokonane w bazie danych w ramach bieżącej transakcji.

Po zatwierdzeniu transakcji zmiany stają się trwałe i widoczne dla innych użytkowników bazy danych.

```
BEGIN TRANSACTION;  
UPDATE Pracownicy SET Stanowisko = 'Kierownik' WHERE ID = 1;  
DELETE FROM Pracownicy WHERE ID = 2;  
COMMIT;
```

TCL – zarządzanie transakcjami

ROLLBACK

Cofnięcie transakcji do stanu sprzed jej rozpoczęcia.

Używane, gdy wystąpi błąd w trakcie transakcji lub użytkownik zdecyduje się anulować wszystkie zmiany dokonane w bazie.

```
BEGIN TRANSACTION;  
UPDATE tabela SET kolumna = 'wartość' WHERE warunek;  
SAVEPOINT nazwa_punktu;  
UPDATE tabela SET kolumna = 'inna wartość' WHERE inny_warunek;  
ROLLBACK TO SAVEPOINT nazwa_punktu;  
COMMIT;
```

SAVEPOINT

Umożliwia utworzenie punktu kontrolnego wewnątrz transakcji, do którego można później cofnąć operacje, bez konieczności anulowania całej transakcji.

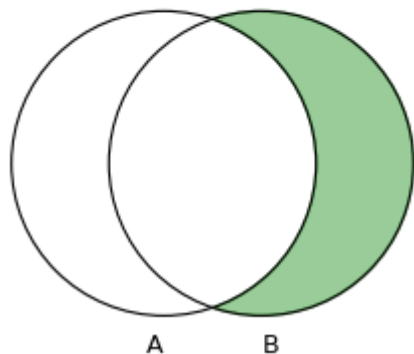
Dzięki SAVEPOINT użytkownik może częściowo anulować operacje, pozostawiając wcześniejsze zmiany.

```
BEGIN TRANSACTION;  
UPDATE Pracownicy SET Stanowisko = 'Kierownik' WHERE ID = 1;  
SAVEPOINT Punkt1; -- Tworzenie punktu kontrolnego  
UPDATE Pracownicy SET Stanowisko = 'Dyrektor' WHERE ID = 2;  
ROLLBACK TO SAVEPOINT Punkt1; -- Cofnięcie do Punkt1, operacja na ID = 2 zostaje cofnięta  
COMMIT; -- Zatwierdzenie pozostałych zmian
```

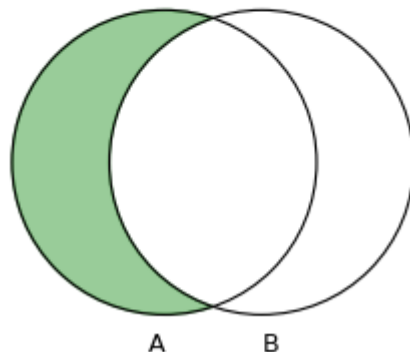
Zapytania SQL – złączenia tabel JOIN



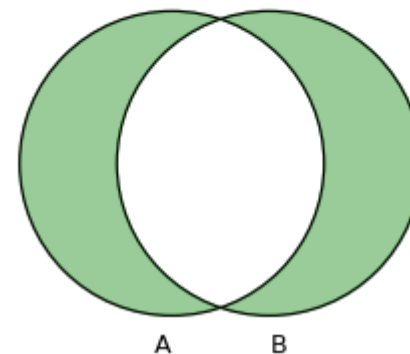
SELECT * FROM A RIGHT JOIN B
ON A.key = B.key
WHERE A.key IS NULL



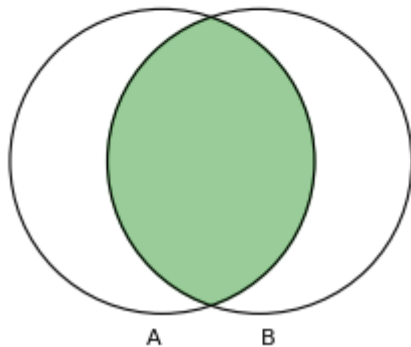
SELECT * FROM A LEFT JOIN B
ON A.key = B.key
WHERE B.key IS NULL



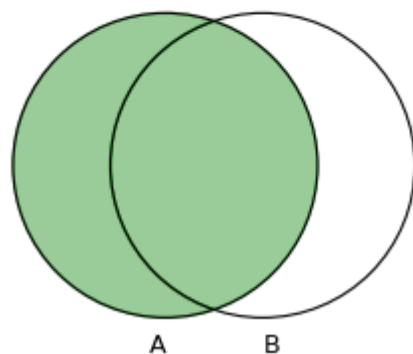
SELECT * FROM A FULL JOIN B
ON A.key = B.key
WHERE A.key IS NULL OR B.key IS NULL



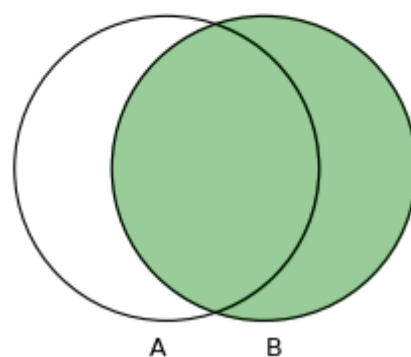
SELECT * FROM A INNER JOIN B
ON A.key = B.key



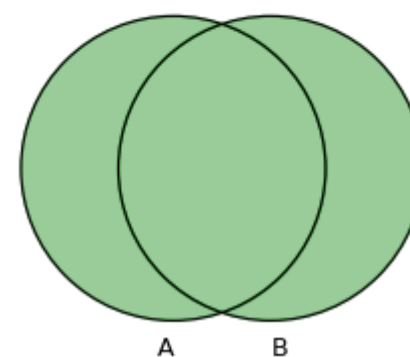
SELECT * FROM A LEFT JOIN B
ON A.key = B.key



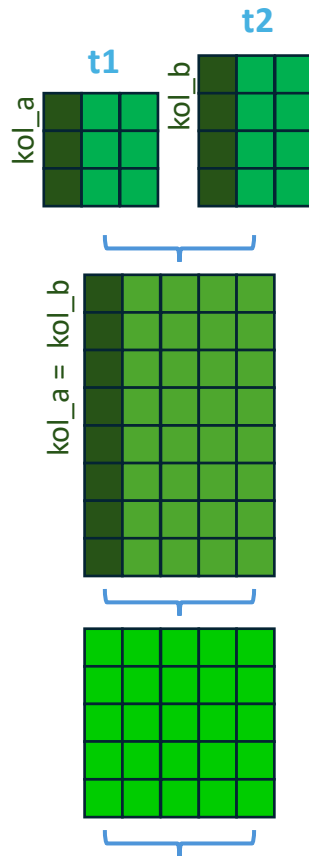
SELECT * FROM A RIGHT JOIN B
ON A.key = B.key



SELECT * FROM A FULL JOIN B
ON A.key = B.key



SQL - SELECT



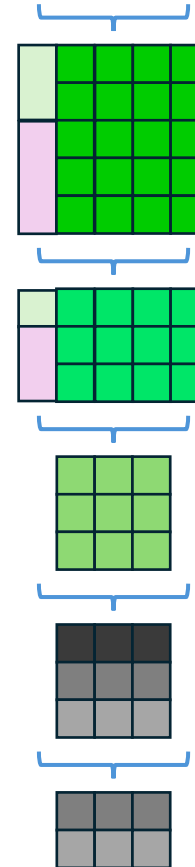
SELECT kol_a, kol_b

FROM t1

JOIN t2

ON t1.kol_a = t2.kol_b

WHERE warunki



GROUP BY kol

HAVING warunki

ORDER BY kol ASC/DESC

LIMIT liczba

Ćwiczenie praktyczne

Treść

Masz bazę danych o nazwie **Firma**, która zawiera tabelę **Pracownicy** z następującą strukturą:

Napisz zapytania SQL, które

- Wyświetl wszystkie dane z tabeli Pracownicy.
- Wyświetl imiona i nazwiska pracowników, którzy pracują w dziale IT.
- Znajdź wszystkich pracowników, którzy zarabiają więcej niż 4000.
- Zaktualizuj pensję Piotra Zielińskiego na 4700.
- Usuń z bazy danych pracownika Adama Malinowskiego.
- Dodaj nowego pracownika o następujących danych: Janina Kowalczyk, Specjalista, Pensja 4200, dział HR.

ID	Imie	Nazwisko	Stanowisko	Pensja	Dział
1	Jan	Kowalski	Manager	5000	Marketing
2	Anna	Nowak	Asystent	3000	HR
3	Piotr	Zieliński	Specjalista	4500	IT
4	Maria	Wiśniewska	Manager	5500	IT
5	Adam	Malinowski	Specjalista	4000	Marketing

Należy wynik przestać w Moodle.

