

## Lekcja

**Temat:** Oddanie i omówienie sprawdzianu

## Lekcja

**Temat:** Zaawansowane zapytania JOIN

```
DROP TABLE IF EXISTS zamowienia;
DROP TABLE IF EXISTS klienci;

CREATE TABLE klienci (
    id INT PRIMARY KEY,
    imie VARCHAR(50)
);

CREATE TABLE zamowienia (
    id INT PRIMARY KEY,
    id_klienta INT ,
    produkt VARCHAR(50),
    FOREIGN KEY (id_klienta) REFERENCES klienci(id)
);

INSERT INTO klienci (id, imie) VALUES
(1, 'Anna'),
(2, 'Jan'),
(3, 'Ola'),
(4, 'Piotr');

INSERT INTO zamowienia (id, id_klienta, produkt) VALUES
(1, 1, 'Laptop'),
(2, 1, 'Myszka'),
(3, 2, 'Telefon'),
(4, null, 'Monitor'); -- ten klient (id=5) nie istnieje w tabeli klienci
```

### INNER JOIN

```
SELECT k.imie, z.produkt
FROM klienci k
INNER JOIN zamowienia z ON k.id = z.id_klienta;
```

imie	produkt
------	---------

Anna	Laptop
Anna	Myszka
Jan	Telefon

## LEFT JOIN

```
SELECT k.imie, z.produkt  
FROM klienci k  
LEFT JOIN zamowienia z ON k.id = z.id_klienta;
```

imie	produkt
Anna	Laptop
Anna	Myszka
Jan	Telefon
Ola	NULL
Piotr	NULL

## RIGHT JOIN

```
SELECT k.imie, z.produkt  
FROM klienci k  
RIGHT JOIN zamowienia z ON k.id = z.id_klienta;
```

imie	produkt
Anna	Laptop
Anna	Myszka
Jan	Telefon
NULL	Monitor

## FULL JOIN (symulowany)

```
SELECT k.imie, z.produkt  
FROM klienci k  
LEFT JOIN zamowienia z ON k.id = z.id_klienta  
  
UNION  
  
SELECT k.imie, z.produkt  
FROM klienci k  
RIGHT JOIN zamowienia z ON k.id = z.id_klienta
```

```
SELECT k.imie, z.produkt
FROM klienci k
RIGHT JOIN zamówienia z ON k.id = z.id_klienta;
```

imie	produkt
Anna	Laptop
Anna	Myszka
Jan	Telefon
Ola	NULL
Piotr	NULL
NULL	Monitor

```
DROP TABLE IF EXISTS zamówienia;
DROP TABLE IF EXISTS produkty;
DROP TABLE IF EXISTS sklepy;

CREATE TABLE sklepy (
    id INT PRIMARY KEY,
    nazwa VARCHAR(50)
);

CREATE TABLE produkty (
    id INT PRIMARY KEY,
    nazwa VARCHAR(50),
    id_sklepu INT,
    FOREIGN KEY (id_sklepu) REFERENCES sklepy(id)
);

CREATE TABLE zamówienia (
    id INT PRIMARY KEY,
    id_produktu INT ,
    ilosc INT,
    FOREIGN KEY (id_produktu) REFERENCES produkty(id)
);

INSERT INTO sklepy (id, nazwa) VALUES
(1, 'Sklep A'),
(2, 'Sklep B'),
(3, 'Sklep C');

INSERT INTO produkty (id, nazwa, id_sklepu) VALUES
(1, 'Laptop', 1),
(2, 'Myszka', 1),
(3, 'Monitor', 2),
```

```
(4, 'Klawiatura', 3);
```

```
INSERT INTO zamówienia (id, id_produktu, ilosc) VALUES  
(1, 1, 5),  
(2, 1, 3),  
(3, 2, 10),  
(4, 3, 2);
```

### Zestawienie sklepów i produktów, łącznie z tymi, dla których nie odnotowano zamówień:

```
SELECT s.nazwa AS sklep,  
       p.nazwa AS produkt,  
       COALESCE(SUM(z.ilosc), 0) AS sprzedane_sztuki  
FROM sklepy s  
LEFT JOIN produkty p ON p.id_sklepu = s.id  
LEFT JOIN zamówienia z ON z.id_produktu = p.id  
GROUP BY s.id, p.id  
ORDER BY s.id, sprzedane_sztuki DESC;
```

sklep	produkt	sprzedane_sztuki
Sklep A	Myszka	10
Sklep A	Laptop	8
Sklep B	Monitor	2
Sklep C	Klawiatura	0

#### Wyjaśnienie:

- LEFT JOIN produkty → bierzemy wszystkie sklepy, nawet jeśli nie mają produktów.
- LEFT JOIN zamówienia → bierzemy wszystkie produkty, nawet jeśli nie mają zamówień.
- SUM(z.ilosc) → sumujemy liczbę sprzedanych sztuk dla każdego produktu.
- COALESCE(..., 0) → jeśli produkt nie ma zamówień, pokazujemy 0 zamiast NULL.
- GROUP BY s.id, p.id → agregujemy dane po sklepie i produkcie.
- ORDER BY s.id, sprzedane\_sztuki DESC → sortujemy dane po sklepie i liczbie sprzedanych sztuk.

### Pokazuje wszystkie zamówienia, nawet jeśli nie ma dopasowanego produktu lub sklepu:

```
SELECT s.nazwa AS sklep,  
       p.nazwa AS produkt,  
       z.ilosc AS sprzedane_sztuki  
FROM sklepy s  
RIGHT JOIN produkty p ON p.id_sklepu = s.id  
RIGHT JOIN zamówienia z ON z.id_produktu = p.id  
GROUP BY s.id, p.id;
```

sklep	produkt	sprzedane_sztuki
Sklep A	Laptop	3

Sklep A	Laptop	5
Sklep A	Myszka	10
Sklep B	Monitor	2

#### **Wyjaśnienie:**

- RIGHT JOIN** produkty p ON p.id\_sklepu = s.id → bierzemy wszystkie produkty, nawet jeśli nie mają sklepu.
- RIGHT JOIN** zamówienia z ON z.id\_produktu = p.id → bierzemy wszystkie zamówienia, nawet jeśli nie mają przypisanego produktu.
- Jeśli w tabeli **produkty** lub **sklepy** brakuje dopasowania → kolumny będą **NULL**.

## **Lekcja**

**Temat:** Funkcje związane z czasem, datą, operatorami łańcuchowymi

#### **Funkcje daty i czasu**

**Link do dokumentacji MySQL:**

<https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html>

Metoda	Wyjaśnienie	Przykład SQL	Wynik
<b>ADDDATE()</b>	Dodaje interwał do daty	<code>SELECT ADDDATE('2024-01-01', INTERVAL 5 DAY);</code>	2024-01-06
<b>ADDTIME()</b>	Dodaje czas	<code>SELECT ADDTIME('10:00:00','02:30:00');</code>	12:30:00
<b>CONVERT_TZ()</b>	Konwersja strefy czasowej	<code>SELECT CONVERT_TZ('2024-01-01 12:00','UTC','Europe/Warsaw');</code>	2024-01-01 13:00
<b>CURDATE()</b>	Bieżąca data	<code>SELECT CURDATE();</code>	2025-11-10
<b>CURTIME()</b>	Bieżący czas	<code>SELECT CURTIME();</code>	np. 14:22:01
<b>DATE()</b>	Zwraca część datową	<code>SELECT DATE('2024-01-01 10:00:00');</code>	2024-01-01
<b>DATE_ADD()</b>	Dodaje interwał do daty	<code>SELECT DATE_ADD('2024-01-01', INTERVAL 1 MONTH);</code>	2024-02-01
<b>DATE_FORMAT()</b>	Formatuje datę	<code>SELECT DATE_FORMAT('2024-01-15','%d-%m-%Y');</code>	15-01-2024
<b>DATE_SUB()</b>	Odejmuje interwał	<code>SELECT DATE_SUB('2024-01-10', INTERVAL 3 DAY);</code>	2024-01-07
<b>DATEDIFF()</b>	Różnica między datami	<code>SELECT DATEDIFF('2024-02-01','2024-01-01');</code>	31

<b>DAY()</b>	Dzień miesiąca	<code>SELECT DAY('2024-01-15');</code>	15
<b>DAYNAME()</b>	Nazwa dnia	<code>SELECT DAYNAME('2024-01-15');</code>	Tuesday
<b>DAYOFMONTH()</b>	Dzień miesiąca	<code>SELECT DAYOFMONTH('2024-01-15');</code>	15
<b>DAYOFWEEK()</b>	Numer dnia tyg. (1=nd)	<code>SELECT DAYOFWEEK('2024-01-15');</code>	3
<b>DAYOFYEAR()</b>	Dzień roku	<code>SELECT DAYOFYEAR('2024-01-15');</code>	15
<b>EXTRACT()</b>	Wyodrębnia część daty	<code>SELECT EXTRACT(YEAR FROM '2024-01-15');</code>	2024
<b>FROM_DAYS()</b>	Dni → data	<code>SELECT FROM_DAYS(750000);</code>	2044-01-22
<b>FROM_UNIXTIME()</b>	UNIX → data	<code>SELECT FROM_UNIXTIME(1700000000);</code>	2023-11-14 22:13:20
<b>HOUR()</b>	Pobiera godzinę	<code>SELECT HOUR('12:45:00');</code>	12
<b>LAST_DAY()</b>	Ostatni dzień miesiąca	<code>SELECT LAST_DAY('2024-02-10');</code>	2024-02-29
<b>MAKEDATE()</b>	Tworzy datę z dnia roku	<code>SELECT MAKEDATE(2024,32);</code>	2024-02-01
<b>MAKETIME()</b>	Tworzy czas	<code>SELECT MAKETIME(10,20,30);</code>	10:20:30
<b>MICROSECOND()</b>	Mikrosekundy	<code>SELECT MICROSECOND('10:00:00.123456');</code>	123456

<b>MINUTE()</b>	Minuta	SELECT MINUTE('12:45:30');	45
<b>MONTH()</b>	Numer miesiąca	SELECT MONTH('2024-05-10');	5
<b>MONTHNAME()</b>	Nazwa miesiąca	SELECT MONTHNAME('2024-05-10');	May
<b>NOW()</b>	Aktualny datetime	SELECT NOW();	2025-11-10 14:20:xx
<b>PERIOD_ADD()</b>	Dodaje miesiące do YYYYMM	SELECT PERIOD_ADD(202401,2);	202403
<b>PERIOD_DIFF()</b>	Ilość miesięcy między okresami	SELECT PERIOD_DIFF(202402,202401);	1
<b>QUARTER()</b>	Kwartał	SELECT QUARTER('2024-05-10');	2
<b>SEC_TO_TIME()</b>	Sekundy → czas	SELECT SEC_TO_TIME(3661);	01:01:01
<b>SECOND()</b>	Sekundy	SELECT SECOND('12:45:59');	59
<b>STR_TO_DATE()</b>	Tekst → data	SELECT STR_TO_DATE('31-01-2024','%d-%m-%Y');	2024-01-31
<b>SUBTIME()</b>	Odejmuje czas	SELECT SUBTIME('10:00:00','01:30:00');	08:30:00
<b>SYSDATE()</b>	Czas wykonania	SELECT SYSDATE();	2025-11-10...

<b>TIME()</b>	Czas z datetime	SELECT TIME('2024-01-01 12:30:45');	12:30:45
<b>TIME_FORMAT()</b>	Formatuje czas	SELECT TIME_FORMAT('12:30:45','%H:%i');	12:30
<b>TIME_TO_SEC()</b>	Czas → sekundy	SELECT TIME_TO_SEC('01:00:00');	3600
<b>TIMEDIFF()</b>	Różnica czasu	SELECT TIMEDIFF('12:00:00','10:00:00');	02:00:00
<b>TIMESTAMP()</b>	Tworzy datetime	SELECT TIMESTAMP('2024-01-01');	2024-01-01 00:00:00
<b>TIMESTAMPADD()</b>	Dodaje interwał	SELECT TIMESTAMPADD(HOUR,2,'2024-01-01 10:00');	2024-01-01 12:00
<b>TIMESTAMPDIFF()</b>	Różnica datetime	SELECT TIMESTAMPDIFF(DAY,'2024-01-01','2024-01-10');	9
<b>TO_DAYS()</b>	Data → dni od roku 0	SELECT TO_DAYS('2024-01-01');	739252
<b>TO_SECONDS()</b>	Data → sekundy od roku 0	SELECT TO_SECONDS('2024-01-01');	64092288000
<b>UNIX_TIMESTAMP()</b>	Aktualny UNIX time	SELECT UNIX_TIMESTAMP();	np. 1768060000
<b>UTC_DATE()</b>	Data UTC	SELECT UTC_DATE();	2025-11-10

<b>UTC_TIME()</b>	Czas UTC	SELECT UTC_TIME();	13:14:xx
<b>UTC_TIMESTAMP()</b>	Datetime UTC	SELECT UTC_TIMESTAMP();	2025-11-10 13:14:xx
<b>WEEK()</b>	Numer tygodnia	SELECT WEEK('2024-01-10');	1
<b>WEEKDAY()</b>	Dzień tyg. (0=pon)	SELECT WEEKDAY('2024-01-10');	3
<b>WEEKOFYEAR()</b>	Tydzien ISO	SELECT WEEKOFYEAR('2024-01-10');	2
<b>YEAR()</b>	Rok	SELECT YEAR('2024-01-10');	2024
<b>YEARWEEK()</b>	Rok + tydzień	SELECT YEARWEEK('2024-01-10');	202402

**UTC (Uniwersalny Czas Koordynowany) to światowy standard czasu atomowego, który służy jako podstawa do ustalania lokalnego czasu w różnych strefach czasowych.** Polska znajduje się w strefie czasowej UTC+1 (czas środkowoeuropejski, CET) zimą i UTC+2 (czas środkowoeuropejski letni, CEST) latem, a lokalny czas w Polsce jest o 1 lub 2 godziny późniejszy od czasu UTC.

- **Co to jest UTC:**
  - UTC to międzynarodowy standard czasu, który jest niezależny od ruchu obrotowego Ziemi i oparty na bardzo precyzyjnym czasie atomowym.

- Jest to punkt odniesienia, taki sam na całym świecie, do którego dodaje się lub od którego odejmuje się czas, aby uzyskać lokalny czas dla danej strefy czasowej.
- **UTC w Polsce:**
  - Polska leży w strefie czasowej UTC+1 (czas zimowy) lub UTC+2 (czas letni).
  - Czas zimowy (CET): Obowiązuje od ostatniej niedzieli października do ostatniej niedzieli marca. Czas lokalny w Polsce jest o 1 godzinę późniejszy niż UTC. (np. jeśli UTC to 12:00, w Polsce jest 13:00).
  - Czas letni (CEST): Obowiązuje od ostatniej niedzieli marca do ostatniej niedzieli października. Czas lokalny w Polsce jest o 2 godziny późniejszy niż UTC. (np. jeśli UTC to 12:00, w Polsce jest 14:00).

#### Zastosowania:

- Programowanie - przechowywanie dat i czasu w bazach danych
- Lotnictwo - koordynacja lotów międzynarodowych
- Internet - synchronizacja serwerów
- Telekomunikacja - koordynacja transmisji
- Nauka - precyzyjne pomiary czasu

**W praktyce:** Gdy widzisz znacznik czasu typu [2025-11-11T14:30:00Z](#), litera "Z" na końcu oznacza właśnie UTC (od "Zulu time" - wojskowego określenia UTC).

#### Przykłady:

- Polska: UTC+1 (zimą) lub UTC+2 (latem)
- Nowy Jork: UTC-5 (zimą) lub UTC-4 (latem)
- Tokio: UTC+9
- Londyn: UTC+0 (zimą) lub UTC+1 (latem)

## Funkcje i operatory łańcuchowe

### Link do dokumentacji MySQL:

<https://dev.mysql.com/doc/refman/8.4/en/string-functions.html>

Metoda	Wyjaśnienie	Przykład	Wynik
<b>ASCII()</b>	Zwraca kod ASCII pierwszego znaku	<code>SELECT ASCII('A');</code>	65
<b>BIN()</b>	Zwraca liczbę w postaci binarnej	<code>SELECT BIN(10);</code>	1010
<b>BIT_LENGTH()</b>	Zwraca długość napisu w bitach	<code>SELECT BIT_LENGTH('ABC');</code>	24
<b>CHAR()</b>	Zwraca znak odpowiadający podanemu kodowi ASCII	<code>SELECT CHAR(65);</code>	'A'
<b>CHAR_LENGTH()</b>	Liczba znaków (nie bajtów)	<code>SELECT CHAR_LENGTH('Łódź');</code>	4
<b>CHARACTER_LENGTH()</b>	To samo co CHAR_LENGTH()	<code>SELECT CHARACTER_LENGTH('Test');</code>	4
<b>CONCAT()</b>	Łączy napisy	<code>SELECT CONCAT('A', 'B', 'C');</code>	'ABC'
<b>CONCAT_WS()</b>	Łączy napisy z separatorem	<code>SELECT CONCAT_WS('-', 'A', 'B', 'C');</code>	'A-B-C'

<b>ELT()</b>	Zwraca element listy na indeksie (1-based)	SELECT ELT(2,'jeden','dwa','trzy');	'dwa'
<b>EXPORT_SET()</b>	Zamienia liczby bitowe na tekst ON/OFF	SELECT EXPORT_SET(5, 'ON', 'OFF', ',', 4);	ON,OFF,ON,OFF
<b>FIELD()</b>	Zwraca pozycję pierwszego argumentu w liście	SELECT FIELD('kot','pies','kot','mysz');	2
<b>FIND_IN_SET()</b>	Pozycja elementu w liście CSV	SELECT FIND_IN_SET('B', 'A,B,C');	2
<b>FORMAT()</b>	Formatuje liczbę z przecinkami	SELECT FORMAT(12345.678, 2);	'12,345.68'
<b>FROM_BASE64()</b>	Dekoduje Base64	SELECT FROM_BASE64('SGVsbG8=');	'Hello'
<b>HEX()</b>	Zamienia liczbę lub tekst na hex	SELECT HEX('ABC');	414243
<b>INSERT()</b>	Wstawia podciąg w podaną pozycję, zastępując określona liczbę znaków	SELECT INSERT('abcdef', 3, 2, 'XYZ');	'abXYZef'
<b>INSTR()</b>	Pozycja pierwszego wystąpienia podciągu	SELECT INSTR('abcabc','ca');	3
<b>LCASE()</b>	To samo co LOWER() – zamienia na małe litery	SELECT LCASE('Test');	'test'

<b>LEFT()</b>	Zwraca określoną liczbę znaków od lewej	SELECT LEFT('abcdef', 3);	'abc'
<b>LENGTH()</b>	Długość napisu w bajtach	SELECT LENGTH('ABC');	3
<b>LIKE</b>	Sprawdza dopasowanie wzorca	SELECT 'Ala' LIKE 'A%';	1
<b>LOAD_FILE()</b>	Wczytuje zawartość pliku (jeśli SQL ma dostęp)	SELECT LOAD_FILE('/path/file.txt');	<i>treść pliku</i>
<b>LOCATE()</b>	Pozycja podciągu (jak INSTR, ale kolejność argumentów odwrotna)	SELECT LOCATE('b','abc');	2
<b>LOWER()</b>	Zamienia na małe litery	SELECT LOWER('TEST');	'test'
<b>LPAD()</b>	Uzupełnia z lewej do zadanej długości	SELECT LPAD('7', 3, '0');	'007'
<b>LTRIM()</b>	Usuwa spacje z lewej	SELECT LTRIM(' test');	'test'
<b>MAKE_SET()</b>	Zwraca listę elementów pasujących do bitów liczby	SELECT MAKE_SET(5,'A','B','C');	'A,C'
<b>MATCH() AGAINST()</b>	Pełnotekstowe wyszukiwanie	SELECT MATCH(text) AGAINST('kot');	<i>ocena dopasowania</i>

<b>MID()</b>	Alias SUBSTRING()	SELECT MID('abcdef', 2, 3);	'bcd'
<b>NOT LIKE</b>	Odwrotność LIKE	SELECT 'Ala' NOT LIKE 'K%';	1
<b>NOT REGEXP</b>	Odwrotność REGEXP	SELECT 'abc' NOT REGEXP '^[0-9]+\$';	1
<b>OCT()</b>	Zamienia liczbę na system ósemkowy	SELECT OCT(15);	'17'
<b>OCTET_LENGTH()</b>	Alias LENGTH()	SELECT OCTET_LENGTH('ABC');	3
<b>ORD()</b>	Kod ASCII pierwszego znaku	SELECT ORD('A');	65
<b>POSITION()</b>	Alias LOCATE()	SELECT POSITION('a' IN 'banan');	2
<b>QUOTE()</b>	Zwraca tekst w bezpiecznej formie (escape)	SELECT QUOTE("Ala's cat");	'Ala\'s cat'
<b>REGEXP</b>	Dopasowanie wyrażenia regularnego	SELECT 'abc123' REGEXP '[0-9]+';	1
<b>REGEXP_INSTR()</b>	Pozycja dopasowania regexu	SELECT REGEXP_INSTR('abc123','[0-9]+');	4
<b>REGEXP_LIKE()</b>	Czy pasuje regex	SELECT REGEXP_LIKE('test123','[a-z]+');	1

<b>REGEXP_REPLACE()</b>	Zamienia dopasowane fragmenty	<pre>SELECT REGEXP_REPLACE('a1b2c3','[0-9]',' X');</pre>	'aXbXcX'
<b>REGEXP_SUBSTR()</b>	Zwraca fragment pasujący do regrexu	<pre>SELECT REGEXP_SUBSTR('abc123','[0-9]+') ;</pre>	'123'
<b>REPEAT()</b>	Powtarza tekst	<pre>SELECT REPEAT('A',3);</pre>	'AAA'
<b>REPLACE()</b>	Podmienia tekst	<pre>SELECT REPLACE('ala ma kota','a','X');</pre>	'XIX mX kotX'
<b>REVERSE()</b>	Odwraza napis	<pre>SELECT REVERSE('kota');</pre>	'atok'
<b>RIGHT()</b>	Znaki od prawej	<pre>SELECT RIGHT('abcdef', 2);</pre>	'ef'
<b>RLIKE</b>	Alias REGEXP	<pre>SELECT 'abc' RLIKE '[a-z]+';</pre>	1
<b>RPAD()</b>	Uzupełnia napis z prawej	<pre>SELECT RPAD('A', 4, '!');</pre>	'A...!'
<b>RTRIM()</b>	Usuwa spacje z prawej	<pre>SELECT RTRIM('test ');</pre>	'test'
<b>SOUNDEX()</b>	Kod fonetyczny słów	<pre>SELECT SOUNDEX('Robert');</pre>	'R163'
<b>SOUNDS LIKE</b>	Porównanie brzmienia	<pre>SELECT 'Robert' SOUNDS LIKE 'Rupert';</pre>	1
<b>SPACE()</b>	Generuje spacje	<pre>SELECT SPACE(5);</pre>	' '

<b>STRCMP()</b>	Porównuje napisy	SELECT STRCMP('abc', 'abd');	-1
<b>SUBSTR()</b>	Podciąg (alias SUBSTRING)	SELECT SUBSTR('abcdef', 2, 3);	'bcd'
<b>SUBSTRING()</b>	Podciąg	SELECT SUBSTRING('abcdef', 3);	'cdef'
<b>SUBSTRING_INDEX()</b>	Podciąg do N-tego separatora	SELECT SUBSTRING_INDEX('a,b,c','','2');	'a,b'
<b>TO_BASE64()</b>	Kodowanie Base64	SELECT TO_BASE64('Hello');	'SGVsbG8='
<b>TRIM()</b>	Usuwa spacje z obu stron	SELECT TRIM(' test ');	'test'
<b>UCASE()</b>	Alias UPPER()	SELECT UCASE('abc');	'ABC'
<b>UNHEX()</b>	Hex → tekst	SELECT UNHEX('414243');	'ABC'
<b>UPPER()</b>	Zamienia na wielkie litery	SELECT UPPER('kot');	'KOT'
<b>WEIGHT_STRING()</b>	Zwraca wewnętrzną wagę znaków (techniczne)	SELECT WEIGHT_STRING('A');	(hex bajty)

**Na następnej lekcji kartkówka z powyższych tabel. Wymagam znajomości metod napisanych czerwonym kolorem**

# Lekcja

## Temat: ERD (Diagram związków encji ang. Entity Relationship Diagram)

**ERD — diagram związków encji**

To graficzny sposób przedstawienia struktury bazy danych:

- jakie **tabele (encje)** istnieją,
- jakie mają **atrybuty (kolumny)**,
- jakie występują **relacje** między tabelami:
  - **1:1**
  - **1:N**
  - **N:M**

ERD jest tworzony zanim powstanie baza danych, aby zaplanować jej strukturę.

**Encja (Entity) = obiekt, który ma znaczenie w systemie i który chcesz zapisać w bazie.**

Inaczej mówiąc:

- 👉 **Encja = tabela w bazie danych**
- 👉 **Atrybut = kolumna w tabeli**

Przykłady encji:

- **User (użytkownik)**
- **Product (produkt)**
- **Order (zamówienie)**
- **Invoice (faktura)**
- **Department (dział firmy)**

Każda encja ma klucz główny (Primary Key, PK) – unikalny identyfikator, np. id.

## Tworzenie krok po kroku diagramu związków encji

### Krok 1: Zidentyfikuj encje (tabele)

### Krok 2: Określ atrybuty

Dla każdej encji określasz pola.

Przykład:

Customer

- id
- first\_name
- last\_name
- email

### Krok 3: Ustal klucze główne

Każda encja ma PK:

### Krok 4: Określ relacje między encjami

1) Relacja 1:1 (One to One)

Jeden rekord odpowiada dokładnie jednemu rekordowi w drugiej tabeli.

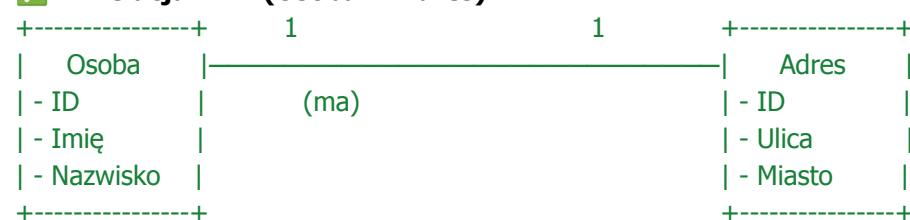
2) Relacja 1:N (One to Many)

Jeden klient może mieć wiele zamówień.

3) Relacja N:M (Many to Many)

Tworzy się tabelę pośredniczącą.

#### 1. Relacja 1 : 1 (Osoba — Adres)

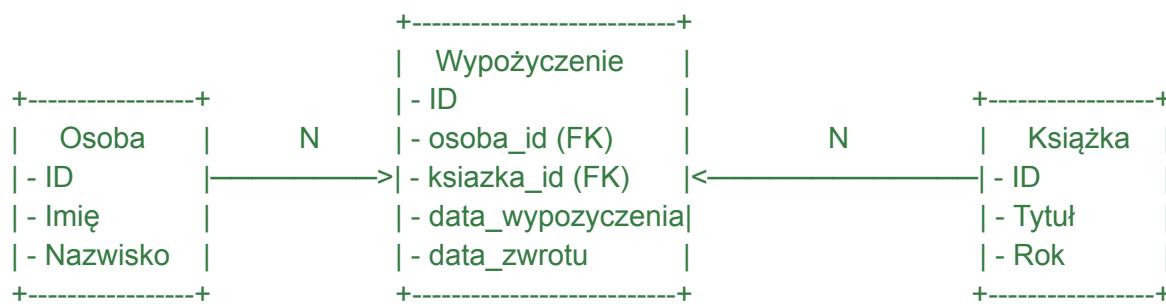


#### 2. Relacja 1 : N (Autor — Książka)



✓ 3. Relacja N : N (Osoba — Książka) przez tabelę Wypożyczenie

W MySQL/SQL relacja N:N **zawsze wymaga tabeli pośredniej**.



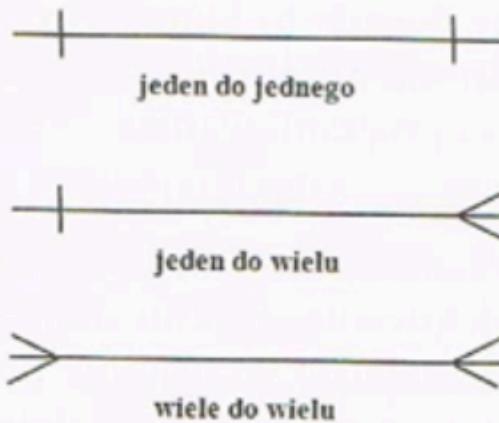
N : N

(wiele osób wypożycza wiele książek)

Opis reprezentacji graficznej stopnia związku został pokazany na rysunku

**Rysunek**

Graficzna reprezentacja związków zachodzących między encjami



Opis reprezentacji graficznej opcjonalności związku został pokazany na rysunku

**Rysunek**

Graficzna reprezentacja opcjonalności związku



Diagramy ERD możemy tworzyć za pomocą różnych notacji. Najpopularniejsze są diagramy w zapisie według Martina i Chena.



**6 osób na następnej lekcji będzie poprawiało sprawdzian**