

PODSTAWY BAZ DANYCH

Materiał przypominający z klasy 3 i 4

Od teorii danych do praktyki języka SQL

Zakres tematyczny:

- Dane, informacja i potrzeba stosowania baz danych
 - Modele i rodzaje baz danych
- Relacyjny model danych i projektowanie ERD
 - Klucze, związki, normalizacja
- Podstawy języka SQL (DDL, DML, DCL, DQL)
 - Transakcje i zasada ACID
- Indeksy, integralność i bezpieczeństwo danych

1. Dane, informacja i potrzeba baz danych

1.1. Dane a informacja

Dana to pojedynczy, surowy fakt — liczba, tekst czy data — który sam w sobie nie niesie jeszcze znaczenia (np. liczba „37”). Informacja powstaje wtedy, gdy dana zostanie umieszczona w kontekście i staje się zrozumiała dla odbiorcy (np. „temperatura powietrza wynosi 37°C”). Zadaniem bazy danych jest przechowywanie danych w taki sposób, aby można było z nich w dowolnym momencie wydobyć potrzebną informację.

1.2. Dlaczego nie wystarczą zwykłe pliki?

Zanim upowszechniły się systemy baz danych, dane przechowywano w odrębnych plikach dla każdej aplikacji (tzw. przetwarzanie plikowe). Rozwiązanie to sprawdzało się w prostych zastosowaniach, ale przy większej skali danych prowadziło do poważnych problemów:

- **nadmiarowość danych** — te same informacje (np. dane klienta) są zapisywane wielokrotnie w różnych plikach,
- **niespójność danych** — zmiana danych w jednym pliku nie jest automatycznie odzwierciedlana w pozostałych,
- **trudny dostęp do danych** — każde nowe zestawienie wymaga napisania osobnego programu,
- **izolacja danych** — dane rozproszone w wielu plikach o różnych formatach trudno ze sobą zestawiać,
- **problemy z integralnością i bezpieczeństwem** — brak scentralizowanej kontroli nad poprawnością i dostępem do danych,
- **problemy ze współbieżnym dostępem** — trudność w bezpiecznej jednoczesnej pracy wielu użytkowników na tych samych danych.

Systemy baz danych powstały właśnie po to, aby rozwiązać powyższe problemy poprzez scentralizowane, uporządkowane i kontrolowane przechowywanie danych.

1.3. Definicja bazy danych i SZBD

Baza danych — uporządkowany, zintegrowany zbiór danych powiązanych ze sobą i dotyczących określonego fragmentu rzeczywistości, przechowywany w sposób umożliwiający łatwy dostęp, przetwarzanie i aktualizację.

System zarządzania bazą danych (SZBD, ang. DBMS — Database Management System) — oprogramowanie pośredniczące między użytkownikiem (lub aplikacją) a bazą danych, które umożliwia tworzenie struktur danych, ich modyfikację, wyszukiwanie oraz zarządzanie bezpieczeństwem i integralnością.

System bazodanowy — całość złożona z bazy danych, SZBD oraz aplikacji korzystających z tych danych.

2. Modele i rodzaje baz danych

Na przestrzeni lat wykształciło się kilka podejść (modeli) do organizacji danych w bazie. Poniższa tabela przedstawia ich chronologiczny rozwój i najważniejsze cechy.

Model danych	Struktura	Najważniejsza cecha / ograniczenie
Kartotekowy (płaski)	pojedyncza tabela / plik	prosty, ale generuje ogromną nadmiarowość danych
Hierarchiczny	drzewo (jeden korzeń, wiele gałęzi)	rekord podrzędny ma dokładnie jednego rodzica — problem z relacjami wiele-do-wielu
Sieciowy	graf (rekord może mieć wielu rodziców)	elastyczniejszy od hierarchicznego, ale skomplikowany w utrzymaniu
Relacyjny	zbiór powiązanych dwuwymiarowych tabel	obecnie dominujący standard; podstawa języka SQL
Obiektowy	obiekty z metodami i dziedziczeniem	łączy bazę danych z programowaniem obiektowym
Obiektowo-relacyjny	tabele rozszerzone o typy obiektowe	hybryda modelu relacyjnego i obiektowego (np. PostgreSQL)
NoSQL	dokumenty, klucz-wartość, grafy, kolumny	elastyczny schemat, wysoka skalowalność (np. MongoDB, Redis)

Uwaga: Współcześnie zdecydowana większość systemów komercyjnych i edukacyjnych opiera się na modelu relacyjnym — dlatego to on jest głównym przedmiotem dalszej części tego opracowania.

3. Relacyjny model danych

3.1. Podstawowe pojęcia

Twórcą relacyjnego modelu danych jest Edgar F. Codd, który opublikował jego założenia w 1970 roku. W modelu tym dane są przechowywane w dwuwymiarowych tabelach, zwanych relacjami, składających się z wierszy i kolumn.

Pojęcie teoretyczne	Odpowiednik praktyczny	Opis
Relacja	Tabela	zbiór danych dotyczących jednego typu obiektów, np. Klienci
Krotka	Rekord / wiersz	pojedynczy zestaw danych, np. dane jednego klienta
Atrybut	Kolumna / pole	pojedyncza cecha obiektu, np. nazwisko, e-mail

Pojęcie teoretyczne	Odpowiednik praktyczny	Opis
Dziedzina	Typ danych	zbiór dopuszczalnych wartości atrybutu, np. liczba całkowita, tekst, data
Stopień relacji	Liczba kolumn	liczba atrybutów w tabeli
Krotność relacji	Liczba wierszy	liczba rekordów aktualnie zapisanych w tabeli

3.2. Reguły poprawnej relacji (tabeli)

- każda tabela ma unikalną nazwę,
- każda kolumna ma unikalną nazwę w obrębie tabeli i przypisany typ danych,
- kolejność kolumn i wierszy nie ma znaczenia,
- każdy wiersz musi być inny — powtarzanie identycznych wierszy jest niedozwolone,
- każda komórka (przecięcie wiersza i kolumny) przechowuje jedną, niepodzielną (atomową) wartość.

3.3. Podstawowe typy danych w SQL

Kategoria	Przykładowe typy	Zastosowanie
Liczby całkowite	INT, SMALLINT, BIGINT	identyfikatory, liczniki, ilości
Liczby zmiennoprzecinkowe	FLOAT, DECIMAL(p,s), NUMERIC	ceny, wartości pieniężne, wyniki pomiarów
Tekst	CHAR(n), VARCHAR(n), TEXT	imiona, nazwiska, opisy, adresy e-mail
Data i czas	DATE, TIME, DATETIME, TIMESTAMP	data urodzenia, data zamówienia
Wartość logiczna	BOOLEAN / BIT	flagi typu tak/nie, aktywny/nieaktywny
Dane binarne	BLOB, VARBINARY	obrazy, pliki, dane multimedialne

W bazach danych (i ogólnie w informatyce) rozróżnia się dwa sposoby zapisu liczb ułamkowych:

Typy stałoprzecinkowe (dokładne)

Przechowują dokładną liczbę cyfr przed i po przecinku — bez błędów zaokrągleń. Używane tam, gdzie liczy się precyzja, np. przy pieniądzach.

Typ SQL	Przykład deklaracji	Przykładowa wartość	Zastosowanie
DECIMAL (p, s)	DECIMAL (10, 2)	1234567.89	ceny, kwoty na fakturach
NUMERIC (p, s)	NUMERIC (6, 3)	123.456	pomiary wymagające stałej precyzji
MONEY (niektóre SZBD, np. SQL Server)	MONEY	2599.99	wartości pieniężne

p (precision) to łączna liczba cyfr, s (scale) to liczba cyfr po przecinku. Np. DECIMAL (10, 2) pomieści liczby do 99999999.99 (8 cyfr przed przecinkiem, 2 po).

Typy zmiennoprzecinkowe (przybliżone)

Przechowują liczbę w postaci mantysy i wykładnika (podobnie jak notacja naukowa), co pozwala zapisać bardzo duże lub bardzo małe wartości, ale kosztem możliwych drobnych błędów zaokrągleń.

Typ SQL	Rozmiar	Przykładowa wartość	Zastosowanie
FLOAT	zwykle 4 bajty (pojedyncza precyzja)	3.14159	dane naukowe, pomiary o mniejszej dokładności
REAL	podobny do FLOAT	98.6	temperatura, wyniki pomiarów
DOUBLE / DOUBLE PRECISION	8 bajtów (podwójna precyzja)	3.14159265358979	obliczenia inżynierskie, GPS, fizyka

Kluczowa różnica na przykładzie

sql

```
-- stałoprzecinkowy — dokładny wynik
SELECT CAST(0.1 AS DECIMAL(10,2)) + CAST(0.2 AS DECIMAL(10,2));
-- wynik: 0.30
```

```
-- zmiennoprzecinkowy — możliwy błąd zaokrąglenia
SELECT CAST(0.1 AS FLOAT) + CAST(0.2 AS FLOAT);
-- wynik: 0.30000000000000004 (w niektórych systemach)
```

Co robi CAST?

CAST to funkcja SQL, która **jawnie zmienia typ danych** wartości na inny, wskazany typ. Ogólna składnia:

sql

```
CAST(wartość AS typ_docelowy)
```

Czyli `CAST(0.1 AS DECIMAL(10,2))` oznacza: „weź wartość 0.1 i potraktuj/zapisz ją jako typ `DECIMAL(10,2)`”, a `CAST(0.1 AS FLOAT)` — „potraktuj tę samą wartość jako typ `FLOAT`”.

Używa się go, gdy chcemy mieć pewność, w jakim typie SZBD wykona obliczenia — bez `CAST` baza mogłaby sama zgadywać typ liczby (np. domyślnie jako `FLOAT`), co utrudniałoby pokazanie różnicy.

Zasada praktyczna: do kwot pieniężnych zawsze używa się typów stałoprzecinkowych (`DECIMAL/NUMERIC`), a typów zmiennoprzecinkowych (`FLOAT/DOUBLE`) — do wartości naukowych i pomiarowych, gdzie niewielki błąd zaokrąglenia nie ma znaczenia.

4. Projektowanie bazy danych — diagram ERD

4.1. Encje, atrybuty i związki

Encja — obiekt lub pojęcie z rzeczywistości, o którym chcemy przechowywać dane (np. Uczeń, Nauczyciel, Zamówienie). Na diagramie reprezentowana jest jako prostokąt i odpowiada tabeli w bazie danych.

Atrybut encji — cecha opisująca encję (np. dla encji Uczeń: imię, nazwisko, data urodzenia). Odpowiada kolumnie tabeli.

Związek (relacja) — logiczne powiązanie między dwiema encjami, np. Uczeń — uczęszcza do — Klasa.

4.2. Klucze

Rodzaj klucza	Skrót	Opis
Klucz główny	PK (Primary Key)	jednoznacznie identyfikuje każdy rekord tabeli; nie może być pusty (NOT NULL) ani powtarzać się
Klucz obcy	FK (Foreign Key)	odwołuje się do klucza głównego innej tabeli i realizuje związek między tabelami
Klucz kandydujący	—	kolumna (lub zestaw kolumn) mogąca pełnić rolę klucza głównego
Klucz złożony	—	klucz zbudowany z więcej niż jednej kolumny

Uwaga: Dobrą praktyką jest tworzenie sztucznego klucza głównego w postaci liczby porządkowej z automatycznym przyrostem (np. `id_ucznia` z właściwością `AUTO_INCREMENT / IDENTITY`), zamiast

wykorzystywania danych rzeczywistych (np. numeru PESEL), które mogą się zmieniać, być niepełne lub nieunikalne.

4.3. Kardynalność (liczebność) związków

Typ związku	Oznaczenie	Przykład	Realizacja w bazie
Jeden do jednego	1:1	Pracownik — Umowa o pracę	klucz obcy w jednej z tabel (rzadko stosowane)
Jeden do wielu	1:N	Klasa — Uczniowie	klucz obcy w tabeli po stronie „wiele”
Wiele do wielu	N:M	Uczniowie — Przedmioty	wymaga tabeli pośredniczącej (łącznikowej) z dwoma kluczami obcymi

4.4. Przykład — od ERD do tabel

Załóżmy prosty system biblioteczny z encjami: Czytelnik, Książka i Wypożyczenie (encja łącznikowa realizująca związek wiele-do-wielu między Czytelnikiem a Książką).

```

CZYTELNIK (id_czytelnika PK, imie, nazwisko, e_mail)
KSIAZKA   (id_ksiazki PK, tytul, autor, rok_wydania)
WYPOZYCZENIE (id_wypozyczenia PK,
               id_czytelnika FK -> CZYTELNIK,
               id_ksiazki    FK -> KSIAZKA,
               data_wypozyczenia, data_zwrotu)

```

Dzięki tabeli WYPOZYCZENIE jeden czytelnik może wypożyczyć wiele książek, a jedna książka może być w przeszłości wypożyczana przez wielu czytelników — związek N:M został poprawnie rozbity na dwa związki 1:N.

5. Normalizacja bazy danych

Normalizacja to proces porządkowania struktury tabel w celu wyeliminowania nadmiarowości danych i związanych z nią anomalii (przy wstawianiu, aktualizacji i usuwaniu danych). Poniżej przedstawiono najważniejsze postacie normalne na praktycznym przykładzie.

Tabela niesznormalizowana

```

ZAMOWIENIE(id_zam, klient, adres_klienta, produkty)
1 | Jan Kowalski | ul. Lipowa 2, Rzeszów | Mysz, Klawiatura

```

Kolumna „produkty” zawiera wiele wartości w jednym polu, co narusza podstawowe zasady modelu relacyjnego.

Pierwsza postać normalna (1NF)

Każda kolumna zawiera wyłącznie wartości atomowe (niepodzielne) — brak list i powtarzających się grup wartości w jednym polu.

```
ZAMOWIENIE(id_zam, klient, adres_klienta, produkt)
1 | Jan Kowalski | ul. Lipowa 2, Rzeszów | Mysz
1 | Jan Kowalski | ul. Lipowa 2, Rzeszów | Klawiatura
```

Druga postać normalna (2NF)

Tabela spełnia 1NF, a każda kolumna niekluczowa zależy od całego klucza głównego (dotyczy kluczy złożonych). Dane klienta nie zależą od produktu, więc wydzielamy osobną tabelę Klienci:

```
KLIENCI(id_klienta PK, klient, adres_klienta)
ZAMOWIENIA(id_zam PK, id_klienta FK)
POZYCJE_ZAMOWIENIA(id_zam FK, produkt)
```

Trzecia postać normalna (3NF)

Tabela spełnia 2NF, a żadna kolumna niekluczowa nie zależy od innej kolumny niekluczowej (zależy wyłącznie od klucza głównego). Przykładowo kod pocztowy nie powinien decydować o nazwie miasta w tej samej tabeli — miasto należy wydzielić do osobnej tabeli słownikowej, jeśli jest powiązane z kodem pocztowym niezależnie od klienta.

Uwaga: W praktyce projektowej dąży się zwykle do 3NF. Wyższe postacie normalne (BCNF, 4NF, 5NF) stosuje się rzadko, w bardziej złożonych przypadkach. Czasem świadomie stosuje się też denormalizację — łączenie tabel z powrotem w celu przyspieszenia odczytu danych kosztem pewnej nadmiarowości.

6. Podstawy języka SQL

SQL (Structured Query Language) to standardowy język służący do definiowania struktur bazy danych, manipulowania danymi oraz zarządzania dostępem do nich. Polecenia SQL dzieli się na cztery główne grupy.

Grupa	Skrót	Przykładowe polecenia	Zastosowanie
Data Definition Language	DDL	CREATE, ALTER, DROP	definiowanie struktury bazy (tabele, klucze, indeksy)
Data Manipulation Language	DML	INSERT, UPDATE, DELETE	wstawianie, modyfikowanie i usuwanie danych
Data Query Language	DQL	SELECT	odczytywanie (wyszukiwanie) danych

Grupa	Skrót	Przykładowe polecenia	Zastosowanie
Data Control Language	DCL	GRANT, REVOKE	nadawanie i odbieranie uprawnień użytkownikom
Transaction Control Language	TCL	COMMIT, ROLLBACK	zatwierdzanie lub wycofywanie transakcji

6.1. Tworzenie tabeli — CREATE TABLE

```
CREATE TABLE Uczniowie (
    id_ucznia    INT PRIMARY KEY AUTO_INCREMENT,
    imie         VARCHAR(50) NOT NULL,
    nazwisko     VARCHAR(50) NOT NULL,
    data_ur      DATE,
    id_klasy     INT,
    FOREIGN KEY (id_klasy) REFERENCES Klasy(id_klasy)
);
```

6.2. Wstawianie, modyfikacja i usuwanie danych

```
INSERT INTO Uczniowie (imie, nazwisko, data_ur, id_klasy)
VALUES ('Anna', 'Nowak', '2008-04-12', 3);
```

```
UPDATE Uczniowie
SET id_klasy = 4
WHERE id_ucznia = 15;
```

```
DELETE FROM Uczniowie
WHERE id_ucznia = 15;
```

6.3. Wyszukiwanie danych — SELECT

```
SELECT imie, nazwisko
FROM Uczniowie
WHERE id_klasy = 3
ORDER BY nazwisko ASC;
```

Klauzula WHERE filtruje wiersze spełniające warunek, ORDER BY porządkuje wynik, a GROUP BY pozwala grupować dane i stosować funkcje agregujące, takie jak COUNT(), SUM(), AVG(), MIN(), MAX().

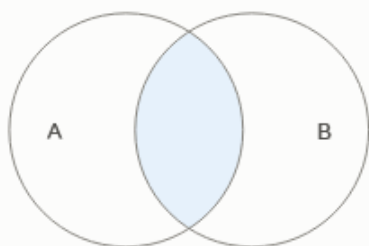
```
SELECT id_klasy, COUNT(*) AS liczba_uczniow
FROM Uczniowie
GROUP BY id_klasy
HAVING COUNT(*) > 20;
```

6.4. Łączenie tabel — JOIN

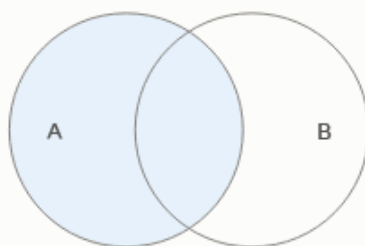
Ponieważ dane w modelu relacyjnym są rozproszone po wielu tabelach, do zestawienia ich w jednym wyniku służy operacja JOIN.

Typ złączenia	Opis
INNER JOIN	zwraca tylko wiersze mające dopasowanie w obu tabelach
LEFT JOIN	zwraca wszystkie wiersze z lewej tabeli, także te bez dopasowania w prawej
RIGHT JOIN	zwraca wszystkie wiersze z prawej tabeli, także te bez dopasowania w lewej
FULL JOIN	zwraca wszystkie wiersze z obu tabel, dopasowane i niedopasowane

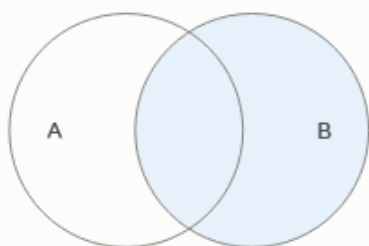
```
SELECT u.imie, u.nazwisko, k.nazwa AS klasa
FROM Uczniowie u
INNER JOIN Klasy k ON u.id_klasy = k.id_klasy;
```

INNER JOIN

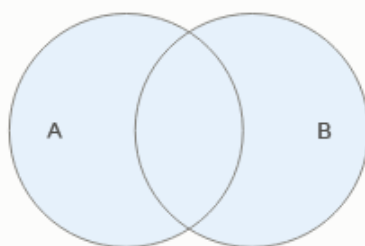
Tylko wiersze pasujące w obu tabelach

LEFT JOIN

Wszystkie wiersze z A + dopasowane z B

RIGHT JOIN

Wszystkie wiersze z B + dopasowane z A

FULL JOIN

Wszystkie wiersze z obu tabel

Zacieniowany obszar = wiersze zwracane przez zapytanie (A = tabela lewa, B = tabela prawa)

7. Transakcje i zasada ACID

Transakcja to ciąg operacji na bazie danych, który powinien zostać wykonany w całości albo wcale — traktowany jest jako jedna, niepodzielna jednostka pracy (np. przelew bankowy: zmniejszenie salda jednego konta i zwiększenie salda drugiego). Poprawność transakcji gwarantują cztery własności określane skrótem ACID:

Litera	Własność	Znaczenie
A	Atomowość (Atomicity)	transakcja wykonuje się w całości albo wcale — nie ma stanów pośrednich
C	Spójność (Consistency)	transakcja przenosi bazę z jednego poprawnego stanu w drugi poprawny stan
I	Izolacja (Isolation)	równolegle wykonywane transakcje nie wpływają na siebie nawzajem
D	Trwałość (Durability)	zatwierdzone (COMMIT) zmiany pozostają w bazie nawet po awarii systemu

```
BEGIN TRANSACTION;
UPDATE Konta SET saldo = saldo - 500 WHERE id_konta = 1;
UPDATE Konta SET saldo = saldo + 500 WHERE id_konta = 2;
COMMIT;
```

8. Indeksy

Indeks to dodatkowa struktura danych przyspieszająca wyszukiwanie rekordów w tabeli — działa podobnie jak skorowidz w książce, pozwalając SZBD ominąć przeglądanie wszystkich wierszy po kolei.

- **zalety:** znacznie szybsze zapytania SELECT i operacje z warunkami WHERE / ORDER BY / JOIN;
- **wady:** indeks zajmuje dodatkowe miejsce na dysku i spowalnia operacje INSERT, UPDATE oraz DELETE, ponieważ musi być na bieżąco aktualizowany.

Klucz główny tabeli otrzymuje indeks automatycznie. Dodatkowe indeksy warto zakładać na kolumnach często używanych w warunkach wyszukiwania, ale nie należy tworzyć ich w nadmiarze (typowo nie więcej niż kilka na jedną tabelę).

```
CREATE INDEX idx_nazwisko ON Uczniowie(nazwisko);
```

9. Integralność i bezpieczeństwo danych

9.1. Rodzaje integralności

- **integralność encji** — każda tabela ma klucz główny, który jednoznacznie i w sposób trwały identyfikuje rekord;
- **integralność referencyjna** — wartości klucza obcego muszą odpowiadać istniejącym wartościom klucza głównego w powiązanej tabeli (lub być NULL, jeśli związek jest opcjonalny);
- **integralność domenowa (semantyczna)** — wartości w kolumnie muszą być zgodne z jej typem danych i dodatkowymi ograniczeniami (np. CHECK, NOT NULL, UNIQUE).

```
CREATE TABLE Produkty (
    id_produktu INT PRIMARY KEY,
    nazwa        VARCHAR(100) NOT NULL,
    cena         DECIMAL(10,2) CHECK (cena > 0),
    kod          VARCHAR(20) UNIQUE
);
```

9.2. Bezpieczeństwo danych

- **uwierzytelnianie** — weryfikacja tożsamości użytkownika (login, hasło);
- **autoryzacja** — nadawanie uprawnień do konkretnych obiektów bazy (tabel, widoków) za pomocą poleceń GRANT / REVOKE;
- **kopie zapasowe (backup)** — regularne tworzenie kopii bazy w celu odtworzenia danych po awarii;
- **szyfrowanie danych** — ochrona danych wrażliwych zarówno podczas przechowywania, jak i przesyłania;
- **dziennik transakcji (log)** — rejestr operacji umożliwiający odtworzenie stanu bazy po awarii.

```
GRANT SELECT, INSERT ON Uczniowie TO uzytkownik_biuro;
REVOKE INSERT ON Uczniowie FROM uzytkownik_biuro;
```

10. Bazy danych NoSQL — wprowadzenie

Bazy NoSQL (ang. „Not Only SQL”) powstały jako odpowiedź na potrzeby przechowywania bardzo dużych, często zmiennych w strukturze zbiorów danych (Big Data), gdzie sztywny schemat relacyjny bywa niewystarczający. W przeciwieństwie do baz relacyjnych, nie wymagają one z góry zdefiniowanego, jednolitego schematu tabel.

Rodzaj bazy NoSQL	Sposób przechowywania danych	Przykładowy system
Dokumentowa	dokumenty JSON / BSON o elastycznej strukturze	MongoDB
Klucz–wartość	proste pary klucz → wartość, bardzo szybki odczyt	Redis
Kolumnowa	dane grupowane kolumnami zamiast wierszami	Cassandra

Rodzaj bazy NoSQL	Sposób przechowywania danych	Przykładowy system
Grafowa	węzły i relacje między nimi (grafy)	Neo4j

Uwaga: Bazy relacyjne i NoSQL nie wykluczają się wzajemnie — w wielu współczesnych systemach stosuje się je równolegle, dobierając odpowiedni typ bazy do charakteru przechowywanych danych.

11. Popularne systemy zarządzania bazami danych

System	Licencja	Typowe zastosowanie
MySQL / MariaDB	open source	aplikacje internetowe, systemy CMS (np. WordPress)
PostgreSQL	open source	zaawansowane aplikacje wymagające zgodności ze standardem SQL
Microsoft SQL Server	komercyjna	systemy korporacyjne w środowisku Windows
Oracle Database	komercyjna	duże systemy korporacyjne i bankowe
SQLite	open source	aplikacje mobilne i desktopowe, lokalne, jednoplikowa baza
MongoDB	open source	aplikacje wymagające elastycznej struktury danych (NoSQL)

12. Słowniczek najważniejszych pojęć

Baza danych — zorganizowany zbiór powiązanych ze sobą danych.

SZBD (DBMS) — oprogramowanie do zarządzania bazą danych.

Tabela (relacja) — zbiór danych o jednym typie obiektów, zorganizowany w wiersze i kolumny.

Rekord (krotka, wiersz) — pojedynczy zestaw danych opisujący jeden obiekt.

Pole (atrybut, kolumna) — pojedyncza cecha obiektu przechowywana w tabeli.

Klucz główny (PK) — kolumna lub zestaw kolumn jednoznacznie identyfikujący rekord.

Klucz obcy (FK) — kolumna odwołująca się do klucza głównego innej tabeli.

Normalizacja — proces eliminowania nadmiarowości danych poprzez podział tabel.

Transakcja — ciąg operacji wykonywanych na bazie danych jako jedna, niepodzielna całość.

Indeks — struktura przyspieszająca wyszukiwanie danych w tabeli.

Zapytanie SQL — polecenie napisane w języku SQL, wysyłane do bazy danych w celu wykonania operacji na danych.

13. Przykładowe zapytania egzaminacyjne (MySQL)

Poniższe cztery zadania to typowe przykłady zapytań pojawiających się w praktycznej części egzaminów zawodowych z kwalifikacji dotyczących baz danych (np. INF.03/EE.09). Łączą podstawowe klauzule SELECT/WHERE z funkcjami wbudowanymi oraz poleceniami DCL.

13.1. Losowanie rekordów z tabeli

Zadanie: wyświetl nazwę produktu i cenę dwóch losowych produktów.

```
SELECT nazwa, cena
FROM Produkty
ORDER BY RAND()
LIMIT 2;
```

RAND() losowo miesza kolejność wierszy, a LIMIT 2 ogranicza wynik do dwóch losowych rekordów.

13.2. Różnica dat i warunek NOT NULL

Zadanie: wyświetl imię i nazwisko zawodnika oraz różnicę w dniach pomiędzy datą bieżącą a datą zapisu (jako kolumnę ilosc_lat_wystepowania), tylko dla zawodników, u których kolumna nagroda nie jest pusta.

```
SELECT imie, nazwisko,
       DATEDIFF(CURDATE(), data_zapisu) AS ilosc_lat_wystepowania
FROM Zawodnicy
WHERE nagroda IS NOT NULL;
```

DATEDIFF(data_do, data_od) zwraca różnicę w dniach między dwiema datami, a CURDATE() zwraca datę bieżącą. Warunek WHERE nagroda IS NOT NULL odrzuca wiersze z pustą wartością w tej kolumnie.

13.3. Warunek na przedziale czasu (BETWEEN)

Zadanie: wyświetl nazwę lotu, miejsce wylotu i miejsce lądowania dla lotów, których czas lądowania mieści się pomiędzy 10:00 a 11:50.

```
SELECT nazwa_lotu, miejsce_wylotu, miejsce_ladowania
FROM Loty
WHERE czas_ladowania BETWEEN '10:00:00' AND '11:50:00';
```

BETWEEN dla kolumny typu TIME działa na zasadzie porównania wartości granicznych włącznie ($\geq$ i $\leq$) — zwróci loty lądujące dokładnie o 10:00, dokładnie o 11:50 oraz w każdej chwili pomiędzy.

13.4. Tworzenie użytkownika i nadawanie uprawnień (DCL)

Zadanie: utwórz użytkownika prezes z hasłem prezes123 na hoście localhost, a następnie nadaj mu uprawnienia do SELECT i UPDATE na tabeli rejsy w bazie danych lotnisko.

```
CREATE USER 'prezes'@'localhost' IDENTIFIED BY 'prezes123';

GRANT SELECT, UPDATE ON lotnisko.rejsy TO 'prezes'@'localhost';

FLUSH PRIVILEGES;
```

CREATE USER tworzy konto ograniczone do połączeń z localhost. GRANT SELECT, UPDATE ON baza.tabela TO 'user'@'host' nadaje dokładnie dwa uprawnienia (odczyt i modyfikację) na jednej konkretnej tabeli, a nie na całej bazie. FLUSH PRIVILEGES odświeża tabelę uprawnień — przy GRANT nie jest to ściśle wymagane, ale to dobry nawyk i często oczekiwany krok na egzaminie.

13.5. Wstawienie rekordu z losową wartością od 1 do 10

```
INSERT INTO samochod (marka, rok_produkcji, waga_tony)
VALUES ('Opel', 2026, FLOOR(1 + RAND() * 10));
```

13.5. Tworzące widok

```
CREATE VIEW widok_ciezkie_samochody AS
SELECT id_samochodu, marka, rok_produkcji, waga_tony
FROM samochod
WHERE waga_tony > 5;
```

(Widok **nie przechowuje własnej kopii danych** — za każdym razem, gdy go odpytujesz, SZBD wykonuje zapisane zapytanie na aktualnych danych z tabeli źródłowej.)

13. Pytania i zadania sprawdzające wiedzę

1. Wyjaśnij różnicę między daną a informacją oraz podaj własny przykład.
2. Wymień co najmniej trzy problemy związane z przechowywaniem danych w plikach zamiast w bazie danych.
3. Na czym polega różnica między kluczem głównym a kluczem obcym? Podaj przykład.
4. Zaprojektuj (opisowo lub w postaci diagramu ERD) bazę danych dla wypożyczalni sprzętu sportowego, uwzględniając co najmniej trzy encje i związki między nimi.
5. Zapisz w języku SQL polecenie tworzące tabelę Pracownicy z kolumnami: id_pracownika (klucz główny), imie, nazwisko, data_zatrudnienia oraz pensja.

6. Wyjaśnij, na czym polega zasada ACID i dlaczego jest istotna przy operacjach bankowych.