# PRZEDMIOT: Systemy baz danych

KLASA: 2A gr. 2

# Tydzień 1 Lekcja 1,2

**Temat**: Definicja Baz Danych. Powtórzenie terminów tabele, rekordy, pola. Relację między tabelami: 1:1, 1:N, N:M. Polecenie Order By. Nadawanie, odbieranie uprawnień (GRANT, REVOKE ). Pojęcie CRUD

## Definicja bazy danych i jej znaczenie:

## **Definicja bazy danych:**

Baza danych to cyfrowy, uporządkowany zbiór informacji, zapisany i przechowywany w sposób ustrukturyzowany, który umożliwia łatwe i szybkie wyszukiwanie, pobieranie, dodawanie, modyfikowanie i usuwanie danych.

## **Znaczenie bazy danych:**

- **Przechowywanie danych** umożliwia gromadzenie dużych ilości informacji w jednym miejscu.
- **Szybki dostęp i wyszukiwanie** dzięki językom zapytań (np. SQL) można błyskawicznie znaleźć potrzebne dane.
- Relacje i spójność pozwala łączyć dane ze sobą (np. klient ↔ zamówienia),
   zachowując integralność.
- **Wielu użytkowników** umożliwia jednoczesną pracę wielu osób/ aplikacji z tymi samymi danymi.
- **Bezpieczeństwo** zapewnia mechanizmy kontroli dostępu i ochrony przed utratą danych.
- Aktualność zmiany wprowadzane w jednym miejscu są natychmiast widoczne dla wszystkich użytkowników.
- **Uniwersalność** używane w niemal każdej dziedzinie (bankowość, handel, medycyna, edukacja, serwisy internetowe).

# Bazy danych można podzielić według sposobu organizacji i przechowywania danych:

•	<ul><li>1. Bazy relacyjne (RDB – Relational Database)</li><li>□ Najpopularniejszy typ.</li></ul>
	☐ Dane są przechowywane w tabelach (wiersze = rekordy, kolumny = pola).
	$\square$ Tabele są powiązane kluczami (np. użytkownik $\rightarrow$ zamówienia).
	□ Do zarządzania używa się języka SQL.
	☐ Przykłady: MySQL, PostgreSQL, Oracle, MS SQL Server.
•	2. Bazy nierelacyjne (NoSQL)
	☐ Dane przechowywane w innych formach niż tabele.
	□ Rodzaje/modele:
	<ul> <li>Dokumentowe dane przechowywane w formie dokumentów (np. JSON, BSON, XML).</li> </ul>
	<ul> <li>Grafowe - dane są przechowywane w postaci grafu (Neo4j – dane jako grafy),</li> </ul>
	<ul> <li>Klucz–wartość - dane przechowywane jako para: klucz →</li> </ul>
	wartość.(Redis, DynamoDB),
	<ul> <li>Kolumnowe - dane zapisane w kolumnach zamiast wierszy</li> </ul>
	(odwrotnie niż w SQL)(Cassandra, HBase).
•	3. Bazy obiektowe
	☐ Dane przechowywane jako <b>obiekty</b> (tak jak w programowaniu obiektowym).
	☐ Mogą przechowywać nie tylko liczby i tekst, ale także multimedia czy złożone
	struktury.
	☐ Przykład: db4o, ObjectDB.
•	4. Bazy obiektowo-relacyjne
	☐ Hybryda relacyjnych i obiektowych.
	☐ Dane przechowywane są w postaci obiektów
	☐ Obsługują tabele, ale także bardziej złożone typy danych.
	☐ Przykład: PostgreSQL, Oracle.
•	5. Bazy hierarchiczne
	☐ Dane są zorganizowane w strukturę <b>drzewa</b> (rodzic–dziecko).
	□ Każdy rekord ma jeden nadrzędny i wiele podrzędnych.
	☐ Szybki dostęp, ale trudne do modyfikacji, mało elastyczne.
	☐ Przykład: IBM IMS (starsze systemy bankowe).

#### 5. Bazy sieciowe

		Dane zorganizowane w strukturze przypominającej <b>sieć</b> lub <b>graf</b> – rekordy mogą mieć wielu rodziców i wielu potomków.
		Stanowią one rozwinięcie modelu hierarchicznego
		Pozwalają na reprezentację danych, gdzie <b>jeden element może być powiązany z wieloma innymi elementami</b> , a <b>te z kolei mogą być</b>
		<b>powiązane z wieloma kolejnymi elementami</b> , tworząc złożoną, grafową strukturę.
		Przykład: IDS (Integrated Data Store).
•	6.	Bazy rozproszone
		Dane nie są przechowywane w jednym miejscu (na jednym serwerze), tylko
		rozsiane po wielu komputerach/serwerach, często w różnych
		lokalizacjach geograficznych.
		Łatwo dodać nowe serwery, gdy rośnie liczba danych.
		Dane są <b>podzielone na części</b> i każda część jest przechowywana na innym
		serwerze pp. użytkownicy A–M są na serwerze 1, a N–Z na serwerze 2.

## Omówienie podstawowych koncepcji: tabele, rekordy, pola

## 📌 1. Tabela

To główna struktura w relacyjnej bazie danych. Można ją porównać do arkusza w Excelu – ma wiersze i kolumny. Każda tabela przechowuje dane dotyczące jednego typu obiektów.

→ Przykład: Tabela Studenci przechowuje informacje o studentach.

## 2. Rekord (wiersz, ang. row/record)

Pojedynczy wiersz w tabeli. Odpowiada jednej jednostce danych (np. jednemu studentowi). Składa się z pól (kolumn).

← Przykład rekordu w tabeli Studenci:

ID Imię Nazwisko Wiek Kierunek1 Anna Kowalska 21 InformatykaTen jeden wiersz to rekord opisujący Annę Kowalską.

# 📌 3. Pole (kolumna, ang. field/column)

#### To kolumna w tabeli, przechowująca określony typ danych.

Każde pole ma nazwę i jest określonego typu danych (np. liczba, tekst, data).

Imię – tekst, Nazwisko – tekst, Wiek – liczba całkowita, Kierunek – tekst.

#### Klucze

# Klucz główny (Primary Key, PK)To unikalny identyfikator rekordu w tabeli.

Gwarantuje, że każdy wiersz można jednoznacznie odróżnić.

Kluczem głównym może być:

- ☐ liczba całkowita (np. ID = 1, 2, 3...),
- ☐ unikalny kod (np. PESEL, NIP),

#### 

ID Imię Nazwisko Wiek1 Anna Kowalska 21

Tutaj ID jest kluczem głównym.

## Klucz obcy (Foreign Key, FK)

To pole w tabeli, które wskazuje na klucz główny w innej tabeli.

Dzięki temu możemy powiązać dane między tabelami.



Tabela Zapisy (które kursy student wybrał) może mieć klucze obce: StudentID  $\rightarrow$  odwołanie do tabeli Studenci(ID), KursID  $\rightarrow$  odwołanie do tabeli Kursy(ID).

#### **Podsumowanie w skrócie:**

- ☐ **Relacyjna baza danych** dane w tabelach powiązane relacjami.
- ☐ **PK** unikalny identyfikator w tabeli.
- ☐ **FK** łączy jedną tabelę z drugą.

# 📌 3. Relacje między tabelami

## 1 Jeden do jednego (1:1)

Każdy rekord w jednej tabeli odpowiada dokładnie jednemu rekordowi w drugiej.

#### **Tabela: Osoby**

id_osoba	imie	nazwisko
1	Adam	Kowalski
2	Anna	Nowak
3	Patryk	Balicki

#### **Tabela: Pesele**

id_pesel	pesel	id_osoby
1	80010112345	1
2	92051267890	2
3	75032145678	3

## 2 Jeden do wielu (1:N)

Jeden rekord w tabeli A może mieć wiele rekordów w tabeli B. Ale rekord w tabeli B należy tylko do jednego w tabeli A.

## Opis relacji

- Jeden nauczyciel może uczyć wiele przedmiotów.
- Ale jeden przedmiot ma przypisanego tylko jednego nauczyciela.

#### **Tabela: Nauczyciele**

id_nauczyciela	imie	nazwisko
1	Adam	Kowalski
2	Anna	Nowak
3	Patryk	Balicki

#### **Tabela: Przedmioty**

id_przedmiotu	nazwa	id_nauczyciela
1	Systemy Baz Danych	1
2	Matematyka	2
3	Fizyka	3
4	Chemia	1

## 3 Wiele do wielu (M:N)

# Rekordy w tabeli A mogą być powiązane z wieloma rekordami w tabeli B i odwrotnie.

### Przykład:

Uczniowie ↔ Przedmioty. Uczeń może zapisać się na wiele przedmiotów, a przedmiot może mieć wielu uczniów.

Rozwiązanie: Tabela Zapisy z polami: id\_ucznia (FK do tabeli Uczniowie) id\_przedmiotu (FK do tabeli Przedmioty ). Trzeba pamiętać, że jednego ucznia nie można przypisać wiele razy do tego samego przedmiotu

#### Tabela: Uczniowie

id_ucznia	imie	nazwisko
1	Adam	Kowalski
2	Anna	Nowak
3	Patryk	Balicki

#### **Tabela: Przedmioty**

id_przedmiotu	nazwa
1	Systemy Baz Danych
2	Matematyka
3	Fizyka
4	Chemia

## Tabela Zapisy (tabela pośrednia)

id_przedmiotu	id_ucznia
1	1
2	1
3	1
2	1
2	2
2	3
3	3
4	1

# Podstawowe polecenia do sortowania

## **★** ORDER BY

```
SELECT nazwisko, imie
FROM pracownicy
ORDER BY nazwisko ASC; -- rosnąco
SELECT nazwisko, imie
FROM pracownicy
ORDER BY nazwisko DESC; -- malejąco
```

## 📌 Sortowanie po wielu kolumnach

```
SELECT nazwisko, imie, pensja
FROM pracownicy
ORDER BY nazwisko ASC, pensja DESC;
```

← Najpierw sortuje po nazwisku rosnąco, a w ramach tego – po pensji malejąco.

### **★ Sortować można** po **numerach kolumn** (niezalecane, ale działa):

```
SELECT nazwisko, imie
FROM pracownicy
ORDER BY 2 ASC, 4 DESC
```

## Zarządzanie bezpieczeństwem bazy danych.

#### Definicje

- **GRANT służy do nadawania uprawnień** użytkownikom bazy danych (np. prawa do odczytu, zapisu, aktualizacji, usuwania, tworzenia tabel).
- REVOKE służy do odbierania wcześniej nadanych uprawnień.

#### Składnia

#### Nadawanie uprawnień (GRANT)

```
GRANT <uprawnienia>
ON <nazwa_bazy_danych>.<nazwa_tabeli>
TO <nazwa_uzytkownika>@<host>;
```

## Odbieranie uprawnień (REVOKE)

```
REVOKE <uprawnienia>
ON <nazwa_bazy_danych>.<nazwa_tabeli>
FROM <nazwa_uzytkownika>@<host>;
```

### **★** CRUD

CRUD to skrót od angielskich słów:

- **C Create** → tworzenie nowych rekordów (np. INSERT)
- **R Read** → odczytywanie danych (np. SELECT)
- **U Update** → aktualizowanie istniejących rekordów (np. UPDATE)
- **D Delete** → usuwanie rekordów (np. DELETE)

### 📌 Odpowiedniki w SQL

- **Create** → INSERT INTO uczniowie (...) VALUES (...)
- **Read** → SELECT \* FROM uczniowie
- **Update** → UPDATE uczniowie SET klasa='3B' WHERE id=1
- **Delete** → DELETE FROM uczniowie WHERE id=1

# Tydzień 2 Lekcja 3,4

**Temat**: Struktura Bazy Danych MySQL

Schemat bazy danych to struktura i organizacja bazy danych, która definiuje jej tabele, pola, relacje, ograniczenia i typy danych. Organizacja baz danych może się różnić od siebie.

## MySQL ma silnik InnoDB.

InnoDB zarządza danymi na własny sposób, korzystając z **tablespace** (**przestrzeni tabel**), które są fizycznymi plikami na dysku. W ich wnętrzu dane są przechowywane w postaci stron (ang. *pages*) i segmentów.

+
TABLESPACE (npibd / ibdata1)
SEGMENT DANYCH (indeks klastrowany = PRIMARY KEY)
Extent #1 (1 MB) — Page (16 KB) → rekordy
Page (16 KB) → rekordy
Extent #2 (1 MB) — Page (16 KB) → rekordy
L
SEGMENT INDEKSU POMOCNICZEGO (np. idx_nazwisko)
SEGMENT UNDO (dla cofania transakcji)
Extent #1 (1 MB) $\rightarrow$ Page (16 KB) $\rightarrow$ wpisy UNDO
L
[inne segmenty/metadane; wolne extenty do przydziału]

```
Tablespace (np. .ibd)
         Extent 1 (1 MB)
| Page (16 KB) | Page (16 KB) | ...
| Rekordy tabeli | Indeksy
| (wiersze) | (np. PRIMARY KEY)|
           Extent 2 (1 MB)
| Page (16 KB) | Page (16 KB) |
| Rekordy tabeli | Undo log / Meta |
            Extent 3 (1 MB)
```

## Gdzie fizycznie są dane tabeli w InnoDB?

### 1. W pliku tablespace:

- jeśli masz innodb\_file\_per\_table=ON (domyślnie) → każda tabela ma własny plik .ibd,
- jeśli innodb\_file\_per\_table=OFF → wszystkie tabele są w system tablespace (ibdata1).

## Rodzaje przestrzeni tabel:

- **System tablespace** główna przestrzeń tabel (zwykle ibdata1) zawierająca metadane, UNDO logi, i ewentualnie dane tabel, jeśli nie korzystasz z trybu "file-per-table".
- **File-per-table tablespace** osobny plik .ibd dla każdej tabeli (jeśli włączone innodb\_file\_per\_table=ON).
- **Temporary tablespaces** dla tabel tymczasowych.
- **Undo tablespaces** do przechowywania danych potrzebnych przy cofnięciu transakcji.

#### Extent

- **Extent** = blok ciągłych stron (*pages*).
- Rozmiar: **1 MB = 64 strony po 16 KB**.
- Extent to czysto fizyczne pojęcie sposób zarządzania przestrzenią w tablespace.
- Extenty są jednostką, którą InnoDB rezerwuje i przydziela tabelom lub indeksom.
- 👉 Możesz to porównać do "klocka" miejsca na dysku.

## Segment

- **Segment** = logiczna struktura w InnoDB, zbudowana z extentów.
- Segmenty są używane do przechowywania różnych rzeczy, np.:
  - segment danych (rekordy tabeli),
  - segment indeksów,
  - segment undo logu.
- Każda tabela w InnoDB ma co najmniej **2 segmenty**:
  - segment danych,
  - segment indeksu klastrowanego (PRIMARY KEY).
- ← Segment to bardziej "pojemnik logiczny", a extenty to jego fizyczne części.

# Relacja segment ↔ extent

- Segment składa się z extentów.
- Extenty zawierają strony (pages).
- Strony zawierają rekordy, indeksy, itp.

## Przykład różnej organizacji systemów baz danych:

W Oracle Blok (block) to Strona (page)

- Oracle
  - **Blok (Block)** = podstawowa jednostka przechowywania danych.
  - Rozmiar bloku w Oracle jest konfigurowalny (np. 2 KB, 4 KB, 8 KB, 16 KB, 32 KB).
  - W bloku są wiersze tabel, wpisy indeksów, nagłówki itd.
- InnoDB (MySQL)
  - **Strona** (**Page**) = podstawowa jednostka przechowywania danych.
  - Rozmiar strony jest prawie zawsze 16 KB (od MySQL 5.7 można zmienić, ale zwykle 16 KB).
  - W stronie są rekordy, sloty, metadane bardzo podobnie jak w bloku Oracle.

### 1. Tworzymy prostą tabelę

```
CREATE TABLE uczniowie (
   id INT PRIMARY KEY AUTO_INCREMENT,
   imie VARCHAR(50),
   nazwisko VARCHAR(50)
) ENGINE=InnoDB;
```

### ← Co powstaje w .ibd?

- segment danych (dla clustered index = PK id),
- segment indeksu klastra.

Czyli minimum 2 segmenty.

## • 2. Dodajemy nowy indeks

ALTER TABLE uczniowie ADD INDEX idx\_nazwisko (nazwisko);

Powstaje nowy segment na ten indeks.

Teraz w pliku .ibd są 3 segmenty:

dane (clustered index),

- indeks klastra,
- dodatkowy indeks idx\_nazwisko.

## • 3. Dodajemy kolumnę typu LOB

ALTER TABLE uczniowie ADD COLUMN opis TEXT;

- → Dla kolumny TEXT tworzony jest osobny segment LOB (Large Object).

  Teraz mamy 4 segmenty:
  - dane,
  - indeks klastra,
  - dodatkowy indeks,
  - segment LOB dla opis.

## 4. Tabela zaczyna rosnąć (np. milion rekordów)

```
INSERT INTO uczniowie (imie, nazwisko, opis)
SELECT 'Jan', 'Kowalski', REPEAT('x', 1000)
FROM generate_series(1, 1000000);
```