Część I – Teoria baz danych (INF.03)

---

1. Rodzaje baz danych

Objaśnienie:

Bazy danych różnią się strukturą i sposobem przechowywania danych:

- **Relacyjne** – dane przechowywane są w tabelach powiązanych ze sobą kluczami (np. MySQL, PostgreSQL, SQLite).
- **Nierelacyjne (NoSQL)** – dane przechowywane są np. jako dokumenty, grafy, pary klucz-wartość (np. MongoDB, Redis).
- **Hierarchiczne** – dane przechowywane są w strukturze drzewa, jak w systemach plików.
- **Sieciowe** – każda jednostka może być powiązana z wieloma innymi (bardziej złożone powiązania niż w relacyjnych bazach).

Zadania:

1. Wypisz po dwa przykłady systemów baz danych dla każdego rodzaju.
2. Podaj przykład zastosowania każdej bazy danych w prawdziwym systemie (np. szkoła, sklep, aplikacja pogodowa).
3. Wskaż, jaki typ bazy będzie najlepszy dla aplikacji społecznościowej i dlaczego.

---

2. Model relacyjny

Objaśnienie:

Model relacyjny zakłada przechowywanie danych w tabelach, złożonych z kolumn (pól) i wierszy (rekordów). Najważniejsze pojęcia:

- **Tabela (relacja)** – zbiór danych o jednym typie obiektu (np. Uczniowie, Zamówienia).
- **Rekord (wiersz)** – pojedynczy wpis w tabeli.
- **Pole (kolumna)** – jedno z danych, np. imię, nazwisko, wiek.
- **Klucz główny (Primary Key)** – jednoznacznie identyfikuje każdy rekord.
- **Klucz obcy (Foreign Key)** – odnosi się do klucza głównego innej tabeli, tworzy relację.

Zadania:

1. Dla tabeli Uczniowie zaprojektuj 5 kolumn, zaznaczając, która będzie kluczem głównym.
2. Narysuj prosty schemat relacyjny dwóch tabel: Klient i Zamówienie, pokazując relację 1:N.
3. Wyjaśnij różnicę między kluczem głównym a obcym i podaj przykłady.

---

3. Normalizacja

Objaśnienie:

Normalizacja to proces upraszczania struktury bazy danych, by:

- unikać powtarzania danych (redukcja redundancji),

- zapewnić spójność danych,
- zwiększyć czytelność.

Najważniejsze postacie normalne:

- **1NF** – każda kolumna zawiera tylko jedną wartość.
- **2NF** – brak częściowej zależności od klucza złożonego.
- **3NF** – brak zależności przechodniej między kolumnami.

Zadania:

1. Dla tabeli Faktury zawierającej kolumny: NumerFaktury, Data, Klient, AdresKlienta, Produkt, Cena – zaproponuj rozdzielenie danych do osobnych tabel (normalizacja).
2. Wyjaśnij, dlaczego przechowywanie adresu klienta w tabeli Faktury może być błędem projektowym.
3. Podaj własny przykład danych, które nie są w 3NF, i popraw je.

---

4. Relacje między tabelami

Objaśnienie:

Relacje określają powiązania między tabelami. Dzięki nim możemy połączyć dane z różnych tabel i utrzymać porządek. Trzy podstawowe typy:

1. **1:1 (jeden do jednego)** – np. Uczeń – Legitymacja (każdy uczeń ma jedną legitymację).
2. **1:N (jeden do wielu)** – np. Klasa – Uczniowie (jedna klasa może mieć wielu uczniów).
3. **N:M (wiele do wielu)** – np. Uczeń – Koło zainteresowań (uczeń może należeć do wielu kół, a koło ma wielu uczniów) – wymaga tabeli pośredniczącej.

Zadania:

1. Podaj po jednym przykładzie relacji 1:1, 1:N i N:M z życia szkoły.
2. Narysuj schemat relacji N:M między tabelami Uczeń i Zajęcia.
3. Dla relacji Nauczyciel – Przedmioty (1:N) zaprojektuj strukturę dwóch tabel i zapisz, jak wygląda klucz obcy.

---

5. Diagram ERD (Entity-Relationship Diagram)

Objaśnienie:

Diagram ERD służy do graficznego przedstawiania struktury bazy danych. Składa się z:

- **Encji** – zaznaczone jako prostokąty (np. Uczeń, Zamówienie)
- **Atrybutów** – owalne (np. imię, data)
- **Relacji** – romby (np. zapisany_do, złożył)
- **Kardynalności** – liczby pokazujące ile rekordów może być powiązanych (np. 1:1, 1:N)

Zadania:

1. Wykonaj diagram ERD dla systemu wypożyczalni filmów z encjami: Klient, Film, Wypożyczenie.
2. Narysuj diagram dla szkoły: Uczeń, Klasa, Nauczyciel, Przedmiot.

3. Zidentyfikuj encje i relacje w tym zdaniu: *"Uczeń wypożycza książki z biblioteki szkolnej i może uczęszczać na kilka zajęć dodatkowych."*

Część II – Projektowanie baz danych

---

1. Tworzenie diagramu encji i relacji dla przykładowego problemu

Objaśnienie:

Zanim stworzysz tabele, najpierw projektujesz logikę systemu:

- **Encje** – jakie obiekty występują w systemie? (np. Uczeń, Klasa)
- **Atrybuty** – jakie dane przechowuje każda encja? (np. imię, nazwisko, wiek)
- **Relacje** – jakie są powiązania między encjami? (np. jeden uczeń należy do jednej klasy)

Diagram ERD pomaga to wszystko narysować i zaplanować.

Zadania:

1. Zaprojektuj diagram encji i relacji dla systemu biblioteki szkolnej.
2. Wymyśl system "Rezerwacja noclegu" – narysuj diagram z encjami: Klient, Pokój, Rezerwacja.
3. Opisz, które atrybuty byłyby obowiązkowe, a które opcjonalne w tabeli Użytkownik systemu ogłoszeniowego.

---

2. Przekształcanie diagramu ERD w strukturę tabel

Objaśnienie:

Po wykonaniu diagramu encji i relacji, każdą **encję** zamieniasz na **tabelę**.

- Atrybuty = kolumny
- Każda tabela musi mieć **klucz główny**
- Relacje = klucze obce

Dla relacji N:M tworzysz **tabelę pośredniczącą**, np. UczeńZajęcia z dwoma kluczami obcymi.

Zadania:

1. Dla diagramu z zadania 1 przekształć encje w tabele – zapisz ich strukturę (nazwy kolumn, typy danych).
2. Zapisz, jakie klucze obce będą potrzebne w systemie Hotel – Rezerwacja – Pokój.
3. Stwórz tabelę pośredniczącą dla relacji N:M Uczeń – KołoZainteresowań.

---

3. Ustalanie kluczy głównych i obcych

Objaśnienie:

- **Klucz główny (PK)** – unikalnie identyfikuje każdy rekord. Zwykle ID, ale może to być też np. PESEL.

- **Klucz obcy (FK)** – pole, które wskazuje na inny rekord z innej tabeli. Dzięki temu powstają relacje między tabelami.

Dobór kluczy ma wpływ na wydajność i spójność danych.

Zadania:

1. W tabeli Pracownik wskaż odpowiedni klucz główny, a w tabeli Zlecenia dodaj klucz obcy wskazujący pracownika.
2. Wymyśl, jaki może być naturalny klucz główny w tabeli Uczniowie (czy zawsze musi być sztuczne ID?).
3. Dla tabel Klient, Zamówienie, Produkt, PozycjaZamówienia – wskaż wszystkie PK i FK.

Część III – SQL: podstawy składni (w XAMPP/phpMyAdmin)

---

1. DDL – Data Definition Language

Objaśnienie:

DDL to język definicji danych – czyli **tworzenie i modyfikowanie struktury tabel**.

Najważniejsze polecenia:

- CREATE TABLE – tworzy nową tabelę
- ALTER TABLE – zmienia istniejącą tabelę
- DROP TABLE – usuwa tabelę
- Klucze i ograniczenia:
    - PRIMARY KEY – unikalny identyfikator rekordu
    - FOREIGN KEY – klucz obcy (łączenie z inną tabelą)
    - NOT NULL – pole wymagane
    - DEFAULT – domyślna wartość

W XAMPP (phpMyAdmin) można:

- pisać zapytania SQL ręcznie (zakładka **SQL**),
- albo tworzyć tabele wizualnie (**Baza danych → Nowa tabela**).

Zadania:

1. Utwórz w phpMyAdmin bazę danych szkola i tabelę uczniowie z kolumnami:
    - id (INT, PRIMARY KEY, AUTO_INCREMENT)
    - imie (VARCHAR(30), NOT NULL)
    - nazwisko (VARCHAR(50))
    - klasa (VARCHAR(5), domyślnie 1A)
2. Za pomocą ALTER TABLE dodaj do tabeli uczniowie kolumnę data_urodzenia typu DATE.
3. Usuń całą tabelę uczniowie za pomocą polecenia DROP TABLE.

---

2. DML – Data Manipulation Language

Objaśnienie:

DML to język manipulowania danymi – czyli **dodawanie, modyfikowanie i usuwanie danych w tabelach**.

- INSERT INTO – dodaje rekord
- UPDATE – modyfikuje istniejące dane
- DELETE – usuwa rekordy

W phpMyAdmin:

- Możesz wpisywać zapytania w zakładce **SQL**
- Lub korzystać z zakładki **Dodaj**, **Przeglądaj**, **Usuń**

Zadania:

1. Dodaj do tabeli uczniowie 3 uczniów:

```
INSERT INTO uczniowie (imie, nazwisko, klasa) VALUES ('Jan', 'Nowak', '2B');
INSERT INTO uczniowie (imie, nazwisko, klasa) VALUES ('Anna', 'Kowalska', '1A');
INSERT INTO uczniowie (imie, nazwisko, klasa) VALUES ('Tomasz', 'Wiśniewski', '3C');
```

2. Zmień klasę ucznia o id = 2 na 2A (UPDATE).
3. Usuń ucznia o nazwisku Wiśniewski z tabeli (DELETE z WHERE).

---

3. DQL – Data Query Language

Objaśnienie:

DQL służy do **pobierania danych** z bazy za pomocą zapytań SELECT. Najważniejsze składniki:

- SELECT * FROM tabela – pobierz wszystkie dane
- WHERE – filtruj wyniki
- ORDER BY – sortuj
- LIMIT – ogranicz ilość wyników
- GROUP BY, HAVING – grupowanie wyników
- Funkcje agregujące: COUNT(), AVG(), SUM(), MAX(), MIN()

W phpMyAdmin najlepiej testować zapytania w zakładce **SQL**.

Zadania:

1. Wyświetl wszystkich uczniów z klasy 2B, posortowanych alfabetycznie po nazwisku.

   ```
   SELECT * FROM uczniowie WHERE klasa = '2B' ORDER BY nazwisko;
   ```

2. Policz, ilu uczniów jest w każdej klasie (GROUP BY klasa + COUNT(*)).
3. Wyświetl tylko tych uczniów, którzy mają nazwisko zaczynające się na literę K (LIKE 'K%').

Część IV – Zaawansowane zapytania SQL (XAMPP / phpMyAdmin)

---

1. Złączenia tabel (JOIN)

Objaśnienie:

JOIN łączy dane z dwóch (lub więcej) tabel na podstawie kluczy. Typy:

- INNER JOIN – zwraca pasujące rekordy z obu tabel
- LEFT JOIN – wszystko z lewej tabeli + pasujące z prawej (reszta = NULL)
- RIGHT JOIN – wszystko z prawej tabeli + pasujące z lewej
- FULL JOIN – łączy wszystko z obu stron (w MySQL trzeba symulować UNION)

Przykład:

Mamy tabele:

- uczniowie(id, imie, nazwisko, klasa_id)
- klasy(id, nazwa, wychowawca)

```
SELECT uczniowie.imie, uczniowie.nazwisko, klasy.nazwa
FROM uczniowie
INNER JOIN klasy ON uczniowie.klasa_id = klasy.id;
```

Zadania:

1. Stwórz tabele klasy i powiąż ją z istniejącą tabelą uczniowie (klasa_id jako FK).
2. Wyświetl listę uczniów wraz z nazwą klasy (INNER JOIN).
3. Użyj LEFT JOIN, by pokazać wszystkie klasy, nawet te bez uczniów.

---

2. Zagnieżdżone zapytania (podzapytania)

Objaśnienie:

Podzapytania (subqueries) to zapytania zagnieżdżone w WHERE, FROM lub SELECT.

Przykład 1 – WHERE:

```
SELECT * FROM uczniowie
WHERE klasa_id = (SELECT id FROM klasy WHERE nazwa = '2A');
Przykład 2 – w SELECT:
SELECT imie, nazwisko,
  (SELECT nazwa FROM klasy WHERE klasy.id = uczniowie.klasa_id) AS klasa
FROM uczniowie;
```

Zadania:

1. Wyświetl wszystkich uczniów z klasy „1B” używając podzapytania.
2. Pokaż imię i nazwisko ucznia oraz nazwę klasy (podzapytanie w SELECT).
3. Napisz zapytanie, które zwraca tylko tych uczniów, którzy są w tej samej klasie co uczeń o nazwisku „Nowak”.

3. Funkcje tekstowe

Objaśnienie:

Funkcje tekstowe pomagają manipulować danymi typu tekst:

- CONCAT(a, b) – łączy tekst
- UPPER(tekst) – zamienia na wielkie litery
- LOWER(tekst) – zamienia na małe litery
- SUBSTRING(tekst, start, długość) – wycina fragment

Zadania:

1. Wyświetl pełne imię i nazwisko ucznia w jednej kolumnie (CONCAT).
2. Pokaż wszystkich uczniów, ale nazwiska zamień na wielkie litery.
3. Pokaż pierwsze 3 litery z nazwiska każdego ucznia.

---

4. Funkcje daty i czasu

Objaśnienie:

Przydatne do pracy z datami:

- NOW() – aktualna data i czas
- YEAR(data) – zwraca rok
- DATEDIFF(data1, data2) – liczba dni między datami
- CURDATE() – tylko data

Zadania:

1. Dodaj do tabeli uczniowie kolumnę data_urodzenia i wprowadź przykładowe dane.
2. Wyświetl uczniów, którzy mają mniej niż 18 lat (YEAR(CURDATE()) - YEAR(data_urodzenia) < 18).
3. Pokaż imię i wiek uczniów (wiek obliczany z daty).

---

5. Tworzenie widoków (VIEW)

Objaśnienie:

Widok (VIEW) to wirtualna tabela na podstawie zapytania SELECT. Ułatwia pracę i ukrywa złożoność.

```
CREATE VIEW uczniowie_z_klasami AS
SELECT u.imie, u.nazwisko, k.nazwa AS klasa
FROM uczniowie u
JOIN klasy k ON u.klasa_id = k.id;
```

Zadania:

1. Utwórz widok pełna_lista_uczniów pokazujący imię, nazwisko i klasę ucznia.
2. Zrób zapytanie SELECT do widoku – jakby to była normalna tabela.
3. Usuń widok (DROP VIEW pełna_lista_uczniów).

6. Grupowanie danych (GROUP BY, HAVING)

Objaśnienie:

GROUP BY służy do **grupowania rekordów według jednej lub wielu kolumn**, a następnie można wykonać **funkcje agregujące** dla każdej grupy:

- COUNT() – liczba rekordów w grupie
- SUM() – suma wartości
- AVG() – średnia
- MAX() / MIN() – największa/najmniejsza wartość
- HAVING – filtr na wynik grupowania (bo WHERE działa przed agregacją)

Przykład:

Policz ilu uczniów jest w każdej klasie:

```
SELECT klasa, COUNT(*) AS liczba_uczniow
FROM uczniowie
GROUP BY klasa;
```

Tylko klasy z więcej niż 3 uczniami:

```
SELECT klasa, COUNT(*) AS liczba_uczniow
FROM uczniowie
GROUP BY klasa
HAVING liczba_uczniow > 3;
```

Zadania:

1. Policz liczbę ogłoszeń przypisanych do każdej kategorii.
2. Policz ilu uczniów jest w każdej klasie (w tabeli uczniowie).
3. Wyświetl tylko te klasy, które mają więcej niż 5 uczniów (HAVING).

---

7. Zarządzanie użytkownikami i uprawnieniami (GRANT, REVOKE)

Objaśnienie:

MySQL pozwala **nadawać i odbierać uprawnienia** użytkownikom bazy. Przykładowe uprawnienia:

- SELECT – może czytać dane
- INSERT, UPDATE, DELETE – może modyfikować dane
- ALL PRIVILEGES – pełny dostęp
- ON database.table – można ograniczyć dostęp do konkretnej bazy/tabeli

W XAMPP/phpMyAdmin:

- Wejdź w zakładkę **Uprawnienia → Dodaj użytkownika**
- Można nadać dostęp tylko do jednej bazy
- Ustaw localhost jako host, hasło własne

Przykład SQL:

```
GRANT SELECT, INSERT ON szkola.* TO 'nowyuser'@'localhost' IDENTIFIED BY 'tajnehaslo';
REVOKE INSERT ON szkola.* FROM 'nowyuser'@'localhost';
```

Zadania:

1. Utwórz nowego użytkownika student1 w phpMyAdmin, z dostępem tylko do bazy szkola (SELECT i INSERT).
2. Nadaj uprawnienia UPDATE dla tabeli uczniowie.
3. Odbierz użytkownikowi prawo do INSERT (REVOKE).

---

8. Funkcje agregujące – objaśnienie

Funkcje agregujące pozwalają **obliczać wartości zbiorcze** (np. sumy, średnie, ilości) dla kolumn. Często występują z GROUP BY, ale mogą też działać na całej tabeli.

Funkcja	Działanie	Przykład
COUNT()	Liczy rekordy	SELECT COUNT(*) FROM uczniowie;
SUM()	Sumuje wartości liczbowe	SELECT SUM(cena) FROM produkty;
AVG()	Średnia wartość	SELECT AVG(wiek) FROM uczniowie;
MIN()	Najniższa wartość	SELECT MIN(wiek) FROM uczniowie;
MAX()	Najwyższa wartość	SELECT MAX(cena) FROM produkty;

Część V – Praktyczne zadania projektowe (w XAMPP/phpMyAdmin)

---

1. Zaprojektuj bazę danych do jednego z tematów:

Objaśnienie:

Zacznij od określenia encji, relacji i pól. Potem przygotuj strukturę tabel i przemyśl:

- które pola będą kluczami głównymi?
- gdzie trzeba wstawić klucz obcy?
- jakie typy danych zastosujesz?

Tematy do wyboru (albo przerób wszystkie):

- **Biblioteka szkolna**
- **Serwis ogłoszeniowy**
- **Rezerwacja noclegów**
- **Sklep internetowy**

Zadania:

1. Wybierz temat i zaprojektuj 3–4 tabele + relacje (np. Użytkownicy, Ogłoszenia, Kategoria).
2. W XAMPP/phpMyAdmin utwórz nową bazę danych i wszystkie tabele z poprawnymi kluczami.
3. Dodaj przykładowe dane do każdej tabeli (przynajmniej 3 rekordy).

---

2. Napisz zestaw 10–15 zapytań SQL do tej bazy

Objaśnienie:

Po utworzeniu struktury i wprowadzeniu danych przejdź do analizy. Pisz zapytania:

- selekcyjne (SELECT)
- modyfikujące (UPDATE, DELETE)
- łączące dane (JOIN)
- z warunkami, sortowaniem, agregacją

Zadania:

1. Napisz zapytanie, które wyświetla np. listę wszystkich ogłoszeń z danej kategorii.
2. Policz, ile ogłoszeń ma każdy użytkownik (GROUP BY).
3. Zmień tytuł ogłoszenia o konkretnym ID (UPDATE).
4. Usuń ogłoszenia starsze niż 3 miesiące (DELETE z warunkiem daty).
5. Wyświetl nazwę użytkownika i treść ogłoszenia (JOIN użytkownicy + ogłoszenia).

---

3. Wersja papierowa i dokumentacja (opcjonalna, ale dobra praktyka)

Objaśnienie:

Jeśli uczysz tego uczniów lub sam chcesz dobrze zrozumieć:

- narysuj diagram ERD (można ręcznie lub np. w dbdiagram.io)
- wypisz strukturę tabel (schematy)
- napisz przykładowe dane
- dopisz kilka zdań: *„Co robi baza?”, „Jakie dane przechowuje?”*

Zadania:

1. Opisz, co robi Twoja baza danych – 5 zdań.
2. Wypisz wszystkie tabele i ich pola + typy danych.
3. Wydrukuj/odrysuj diagram ERD.