PRZEDMIOT: Baz danych

KLASA: 5i 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. Nadawanie, odbieranie uprawnień (GRANT, REVOKE)

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).

	zy danych można podzielić według sposobu organizacji i
pr	zechowywania danych:
•	1. Bazy relacyjne (RDB – Relational Database)
	□ Najpopularniejszy typ.
	☐ Dane są przechowywane w tabelach (wiersze = rekordy, kolumny = pola).
	☐ Tabele są powiązane kluczami (np. użytkownik → 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:
	 Dokumentowe dane przechowywane w formie dokumentów (np. JSON, BSON, XML).
	 Grafowe - dane są przechowywane w postaci grafu (Neo4j – dane jako grafy),
	 ○ Klucz-wartość - dane przechowywane jako para: klucz → wartość.(Redis, DynamoDB),
	 Kolumnowe - dane zapisane w kolumnach zamiast wierszy
	(odwrotnie niż w SQL)(Cassandra, HBase).
•	3. Bazy obiektowe
	 □ Dane przechowywane jako obiekty (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 drzewa (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 sieć lub graf − rekordy mogą mieć wielu rodziców i wielu potomków.
☐ Stanowią one rozwinięcie modelu hierarchicznego
□ Pozwalają na reprezentację danych, gdzie jeden element może być
powiązany z wieloma innymi elementami, a te z kolei mogą być
powiązane z wieloma kolejnymi elementami, tworząc złożoną, grafowa
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 podzielone na części i każda część jest przechowywana na innym

Tworzenie nowej bazy:

1 Microsoft Access

- Access jest bazą plikową, więc baza danych to plik .accdb.
- Tworzenie bazy odbywa się **graficznie**, ale można też użyć SQL do tworzenia tabel w już otwartym pliku.
- - 1. Otwórz Access \rightarrow Plik \rightarrow Nowy \rightarrow Pusta baza danych
 - 2. Nadaj nazwę, np. Sklep.accdb
 - 3. Access utworzy plik bazy danych i otworzy pustą bazę.

W Access SQL nie ma polecenia typu CREATE DATABASE, bo baza to plik.

SQL w Access służy głównie do tworzenia tabel, zapytań, widoków.

2 PostgreSQL

- PostgreSQL jest serwerową bazą danych.
- Tworzenie bazy odbywa się komendą CREATE DATABASE.

Przykład:

CREATE DATABASE sklep
WITH
OWNER = postgres
ENCODING = 'UTF8'
LC_COLLATE = 'pl_PL.UTF-8'
LC_CTYPE = 'pl_PL.UTF-8'
TEMPLATE = template0;

- OWNER właściciel bazy (użytkownik PostgreSQL)
- ENCODING kodowanie znaków
- LC_COLLATE i LC_CTYPE lokalizacja i sortowanie znaków
- TEMPLATE szablon bazy (zwykle template0 lub template1)

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 Kierunek 1 Anna Kowalska 21 Informatyka Ten 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).

Przykłady pól w tabeli Studenci:

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:

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

W tabeli Studenci:

Nazwisko ID Imię Wiek 1 21 Anna Kowalska

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.

Przykład:

Tabela Zapisy (które kursy student wybrał) może mieć klucze obce:

StudentID → 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: Pesel

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.

← Przykład: Nauczyciele ↔ Przedmioty. Jeden nauczyciel prowadzi wiele przedmiotów, ale każdy przedmiot ma tylko jednego nauczyciela.

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.

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 <baza>.<obiekt>
TO <użytkownik>@<host>;

Odbieranie uprawnień (REVOKE)

REVOKE <uprawnienia>
ON <baza>.<obiekt>
FROM <użytkownik>@<host>;