# Czym są bazy danych i po co są potrzebne?

Wyobraź sobie, że prowadzisz sklep i musisz zapisywać informacje o swoich klientach, produktach i zamówieniach. Możesz robić to na kartce papieru lub w Excelu, ale gdy tych informacji zrobi się bardzo dużo, ciężko będzie je odnaleźć, uporządkować czy przetwarzać.

Bazy danych to specjalne miejsca do przechowywania takich informacji w sposób zorganizowany, aby można było je łatwo znaleźć, zmieniać lub analizować.

Dzięki bazom danych:

- Możesz szybko znaleźć potrzebne informacje (np. listę klientów z Warszawy).

- Dane są bezpieczne i uporządkowane.

- Automatyzujesz codzienną pracę, np. wystawianie faktur czy analizę sprzedaży.

# Co to są relacyjne bazy danych?

Relacyjne bazy danych to jeden z najpopularniejszych rodzajów baz.

Informacje są przechowywane w tabelach, które przypominają arkusz kalkulacyjny (jak w excelu):

Kolumny opisują rodzaje danych (np. imię, nazwisko, adres). Wiersze to poszczególne wpisy (np. konkretna osoba lub zamówienie). Tabele w relacyjnych bazach danych mogą się ze sobą łączyć.

Na przykład:

Tabela Klienci może zawierać dane o osobach (imię, nazwisko).

Tabela Zamówienia może zawierać dane o zakupach, które wykonali ci klienci.

1. Wstawianie danych do bazy (INSERT)

Załóżmy, że mamy tabelę o nazwie klienci, z kolumnami:

|  |  |  |
| --- | --- | --- |
| id (numer klienta) | imie (imię klienta), | nazwisko (nazwisko ) |

Aby dodać nowego klienta:

INSERT INTO klienci (id, imie, nazwisko)

VALUES (1, 'Jan', 'Kowalski');

2. Pobieranie danych z bazy (SELECT)

Jeśli chcesz zobaczyć wszystkich klientów:

SELECT \* FROM klienci;

To polecenie oznacza:

Pokaż wszystkie wiersze i wszystkie kolumny z tabeli klienci.

Jeśli chcesz znaleźć klientów o nazwisku "Kowalski":

SELECT \* FROM klienci **WHERE nazwisko = 'Kowalski';**

To polecenie oznacza:

Pokaż tylko te wiersze, gdzie nazwisko to "Kowalski".

# Co to jest JOIN w bazach danych?

JOIN to sposób łączenia danych z różnych tabel w bazie danych. Wyobraź sobie, że masz dwie tabele:

1. **klienci** – przechowuje dane o klientach (np. ich imiona i nazwiska).
2. **zamowienia** – przechowuje informacje o zamówieniach złożonych przez tych klientów.

Tabela **klienci** może wyglądać tak:

|  |  |  |
| --- | --- | --- |
| id (numer klienta) | imie (imię klienta), | nazwisko (nazwisko ) |
| 1 | Jan | Kowalski |
| 2 | Anna | Nowak |

Tabela **zamowienia** może wyglądać tak:

|  |  |  |
| --- | --- | --- |
| id | Klient\_id (id klienta), | produkt |
| 1 | 1 | Telefon |
| 2 | 1 | Laptop |
| 3 | 2 | Tablet |

Kolumna **klient\_id** w tabeli **zamowienia** mówi, który klient złożył dane zamówienie. Teraz chcemy połączyć te dane, aby zobaczyć imiona i nazwiska klientów wraz z ich zamówieniami.

**Przykład JOIN**

Używamy INNER JOIN, aby połączyć te dwie tabele.

SELECT klienci.imie, klienci.nazwisko, zamowienia.produkt

FROM klienci

INNER JOIN zamowienia

ON klienci.id = zamowienia.klient\_id;

**Co robi to zapytanie?**

1. Wybiera kolumny imie i nazwisko z tabeli **klienci** oraz produkt z tabeli **zamowienia**.
2. Łączy wiersze z tabeli **klienci** z wierszami z tabeli **zamowienia**, tam gdzie wartość id w tabeli **klienci** jest równa wartości klient\_id w tabeli **zamowienia**.

**Wynik zapytania**

|  |  |  |
| --- | --- | --- |
| id (numer klienta) | Nazwisko | produkt |
| Jan | Kowalski | Telefon |
| Jan | Kowalski | Laptop |
| Anna | Nowak | Tablet |

(Zielone pochodzą z tabeli klient, pomarańczowe z tabeli produkt a widzisz je zestawione razem, bo połączyliśmy je poleceniem JOIN!)

**Co widzimy?**

* Jan Kowalski złożył dwa zamówienia: Telefon i Laptop.
* Anna Nowak złożyła jedno zamówienie: Tablet.

**Dlaczego JOIN jest przydatny?**

* Pozwala powiązać dane z różnych tabel, co jest niezbędne w dużych bazach danych.
* Dzięki JOIN możesz tworzyć bardziej zaawansowane analizy, np. śledzić historię zamówień klientów, analizować sprzedaż, itp.

Prosty JOIN, jak w powyższym przykładzie, to podstawowe narzędzie w pracy z relacyjnymi bazami danych!

# Co to są klucze główne i obce w bazach danych?

Aby zrozumieć, jak dane w relacyjnych bazach danych są ze sobą powiązane, trzeba poznać pojęcia **klucza głównego (primary key)** i **klucza obcego (foreign key)**. Te klucze to sposób, w jaki baza danych „wie”, które dane są powiązane.

**Klucz główny (Primary Key)**

* **Co to jest?**  
  Klucz główny to unikalny identyfikator dla każdego wiersza w tabeli.  
  **Każda tabela powinna mieć klucz główny.**  
  Dzięki temu można jednoznacznie wskazać konkretny wiersz.

|  |  |  |
| --- | --- | --- |
|  |  |  |

W tej tabeli klienci kolumna id jest **kluczem głównym**, ponieważ każdy klient ma unikalny numer (1, 2 itd.).

**Klucz obcy (Foreign Key)**

|  |  |  |
| --- | --- | --- |
| **Co to jest?** Klucz obcy to kolumna w jednej tabeli, która wskazuje na klucz główny w innej tabeli. Dzięki temu można połączyć dane między tabelami. |  |  |

Kolumna klient\_id jest **kluczem obcym**, ponieważ wskazuje na kolumnę id z tabeli klienci. Dzięki temu wiemy, które zamówienia należą do którego klienta:

* klient\_id = 1 wskazuje na Jana Kowalskiego,
* klient\_id = 2 wskazuje na Annę Nowak.

**Dlaczego klucze są ważne?**

1. **Klucz główny:** Gwarantuje, że każdy wiersz w tabeli jest unikalny i łatwy do zidentyfikowania.
2. **Klucz obcy:** Pozwala łączyć dane z różnych tabel, tworząc relacje między nimi (np. kto zamówił co).

# Jak zdefiniować klucz główny i obcy w PostgreSQL?

**Tworzenie tabeli z kluczem głównym:**

CREATE TABLE klienci (

id SERIAL **PRIMARY KEY,**

imie VARCHAR(50),

nazwisko VARCHAR(50)

);

* Kolumna id jest kluczem głównym dzięki PRIMARY KEY.
* SERIAL oznacza, że wartości w tej kolumnie będą automatycznie numerowane (1, 2, 3...).

**Tworzenie tabeli z kluczem obcym:**

CREATE TABLE zamowienia (

id SERIAL PRIMARY KEY,

klient\_id INT **REFERENCES** klienci(id),

produkt VARCHAR(100)

);

* Kolumna klient\_id jest kluczem obcym dzięki REFERENCES klienci(id), co oznacza, że musi wskazywać na istniejący id w tabeli klienci.

# Przykład z JOIN i kluczami

Kiedy używamy JOIN, klucz obcy pozwala powiązać dane między tabelami.

SELECT klienci.imie, klienci.nazwisko, zamowienia.produkt

FROM klienci

INNER JOIN zamowienia

ON klienci.id = zamowienia.klient\_id;

Wynik łączy dane na podstawie relacji **klucz główny → klucz obcy**.

**Podsumowanie**

1. **Klucz główny (Primary Key)**: Unikalny identyfikator każdego wiersza w tabeli.
2. **Klucz obcy (Foreign Key)**: Łączy tabele, wskazując na klucz główny w innej tabeli.
3. Klucze pomagają zachować porządek i umożliwiają tworzenie relacji między danymi w tabelach.