Bazy danych

Implementacja modelu danych

Dr inż. Arkadiusz Rzucidło, prof. PRz

Implementacja modelu danych
Tworzenie tabel na podstawie encji

„to co analizowaliśmy → zamienia się w prawdziwą bazę danych”

- Encja → tabela
- Pacjent → PACJENT
- Lekarz → LEKARZ
- Wizyta → WIZYTA
- Recepta → RECEPTA
- Lek → LEK

Implementacja modelu danych
Tworzenie tabel na podstawie encji

- To pierwszy moment, gdzie robimy realne przejście:
  - z analizy → do bazy danych
- Każda encja, którą zidentyfikowaliśmy, staje się:
  - **tabelą w bazie danych**

Czyli:

  - obiekt Pacjent → tabela PACJENT
  - obiekt Lekarz → tabela LEKARZ
- To bardzo ważna zasada:
  - jedna encja = jedna tabela
- Jeśli analizę wykonano poprzwnie, to projekt tabel powstaje niemal automatycznie.

Implementacja modelu danych
Tworzenie tabel na podstawie encji

Co powstaje z encji w bazie danych?

A raport
B formularz
C zapytanie
D tabela

Tabela PACJENT

- PACJENT:
  - ID_pacjenta
  - imię
  - nazwisko
  - PESEL
  - adres
  - telefon

Tabela PACJENT

- Zaczynamy od pierwszej tabeli: PACJENT
- To jedna z najprostszych tabel, bo:
  - nie zależy od innych tabel
  - zawiera dane podstawowe
- Każdy rekord w tej tabeli to:
  - jeden pacjent
- Najważniejszy element:
  - ID_pacjenta (klucz główny)
- To on będzie używany do łączenia danych w systemie.

Tabela PACJENT

Które pole powinno być kluczem głównym?

A imię
B nazwisko
C PESEL
D ID_pacjenta

Tabela LEKARZ

- LEKARZ:
  - ID_lekarza
  - imię
  - nazwisko
  - specjalizacja
  - numer_PWZ

Tabela LEKARZ

- Druga tabela: LEKARZ
- Zasady są identyczne:
  - każdy lekarz = jeden rekord
  - posiada własny identyfikator
- Dlaczego nie używamy np. nazwiska jako klucza?
  - bo może się powtarzać
- Dlatego zawsze stosujemy:
  - sztuczny identyfikator (ID)

Tabela LEKARZ

Dlaczego nazwisko nie jest dobrym kluczem głównym?

A jest za krótkie
B jest trudne do zapisania
C może się powtarzać
D nie ma znaczenia

Tabela WIZYTA

- WIZYTA:
  - ID_wizyty
  - data
  - godzina
  - ID_pacjenta
  - ID_lekarza
  - rozpoznanie

Tabela WIZYTA

- To najważniejsza tabela w systemie.
- Dlaczego? bo łączy inne tabele
- WIZYTA zawiera:
  - ID_pacjenta
  - ID_lekarza
- To są:
  - **klucze obce**
- Dzięki nim:
  - wiemy który pacjent ma wizytę
  - wiemy który lekarz ją prowadzi
- To dokładnie odwzorowuje relacje z wykładu 2.

Tabela WIZYTA

Co reprezentują pola ID_pacjenta i ID_lekarza?

A klucze główne
B atrybuty tekstowe
C indeksy
D klucze obce

Tabela RECEPTA

- Tabela RECEPTA jest powiązana z wizytą.
- Każda recepta:
  - należy do jednej wizyty
- Dlatego zawiera:
  - ID_wizyty
- To jest kolejny przykład:
  - relacja → klucz obcy
- Struktura bazy powstaje bezpośrednio z analizy.

Tabela RECEPTA

Z czym powiązana jest recepta?

A pacjentem
B lekarzem
C lekiem
D wizytą

Tabela LEK (szczegóły recepty)

- LEK:
  - ID_leku
  - nazwa
  - dawka

Tabela LEK

- Tabela LEK przechowuje informacje o lekach.
- Ale pojawia się ważne pytanie:
  - jak powiązać leki z receptą?
- Bo:
  - jedna recepta → wiele leków
  - jeden lek → może być na wielu receptach
- To oznacza relację:
  - **wiele do wielu (N:N)**
- A to oznacza:
  - potrzebna będzie dodatkowa tabela (wyjaśni się to później)

Tabela LEK

Jaką relację mają recepta i lek?

A N:N
B 1:N
C 0:1
D 1:1

Efekt dydaktyczny

- Widać jak powstają tabele
- Widać do czego służą klucze główne
- Widać zasadność użycia kluczy obcych
- Można w ten sposób zrozumieć powiązanie z analizą

Klucze i relacje
Klucz główny (Primary Key)

- Cel: zrozumienie **Primary Key, Foreign Key i relacji między tabelami**

- **Primary Key (PK)**
  - unikalny identyfikator rekordu
  - nie może się powtarzać
  - nie może być pusty

Przykład:

  - ID_pacjenta

Klucze i relacje
Klucz główny (Primary Key)

- Każda tabela w bazie danych musi mieć coś, co pozwala:
  - jednoznacznie zidentyfikować każdy rekord
- To właśnie:
  - **klucz główny (Primary Key)**

Przykład:

- Tabela PACJENT:
  - Jan Kowalski
  - Jan Kowalski

Klucze i relacje
Klucz główny (Primary Key)

- Dwie osoby mogą mieć takie same dane.
- Ale:
  - ID_pacjenta jest zawsze unikalne
- To jest fundament każdej bazy danych.

Klucze i relacje
Klucz główny (Primary Key)

Która cecha najlepiej opisuje klucz główny?

A może się powtarzać
B jest unikalny
C jest opcjonalny
D może być pusty

Klucz obcy (Foreign Key)

Która cecha najlepiej opisuje klucz główny?

- **Foreign Key (FK)**
  - wskazuje rekord w innej tabeli
  - tworzy relacje między tabelami

Przykład:

- ID_pacjenta w tabeli WIZYTA

Klucz obcy (Foreign Key)

- Klucz obcy to:
  - pole, które wskazuje na rekord w innej tabeli

Przykład:

- Tabela WIZYTA zawiera:
  - ID_pacjenta
- To oznacza:
  - każda wizyta jest przypisana do konkretnego pacjenta
- To jest dokładnie odwzorowanie relacji:
  - pacjent → wizyta

Klucz obcy (Foreign Key)

Do czego służy klucz obcy?

A przechowywania tekstu
B identyfikacji rekordu
C tworzenia raportów
D łączenia tabel

Relacje między tabelami

- Relacje w bazie:
  - PACJENT → WIZYTA
  - LEKARZ → WIZYTA
  - WIZYTA → RECEPTA

- Relacje realizowane przez:
  - klucze obce

Relacje między tabelami

- To jest moment, w którym baza danych zaczyna **„żyć”.**
- Nie mamy już oddzielnych tabel — mamy:
  - **system powiązanych danych**
- Relacje tworzą:
  - historię pacjenta
  - powiązania lekarz–pacjent
  - przepływ informacji
- Bez relacji baza danych byłaby tylko zbiorem tabel.

Relacje między tabelami

Co łączy tabele w bazie danych?

A raporty
B formularze
C klucze obce
D zapytania

Typy relacji

- Typy relacji:
  - 1 : 1
  - 1 : N
  - N : N
- Najczęstsza:
  - 1 : N

Typy relacji

- W bazach danych wyróżniamy trzy główne typy relacji:
- 1:1 – rzadko stosowane
- 1:N – najczęściej stosowane
- N:N – wymagają dodatkowej tabeli

Przykład:

- Pacjent → Wizyta → 1:N
- Recepta → Lek → N:N
- Relacje N:N nie mogą być zapisane bezpośrednio — wymagają dodatkowej tabeli.

Typy relacji

Która relacja występuje najczęściej?

A 1:N
B N:N
C 0:1
D 1:1

Integralność danych

- Integralność danych oznacza:
  - spójność danych
  - poprawność relacji
  - brak „osieroconych” rekordów

Integralność danych

- To jeden z najważniejszych aspektów baz danych.
  - **integralność danych**
- Oznacza, że:
  - nie możemy mieć wizyty bez pacjenta
  - nie możemy mieć recepty bez wizyty
- DBMS pilnuje tego automatycznie.

Przykład:

- Nie można usunąć pacjenta, jeśli ma wizyty — chyba że obsłużymy to poprawnie.
- To chroni system przed błędami.

Integralność danych

Co oznacza integralność danych?

A szybkość działania
B spójność i poprawność danych
C ilość danych
D wygląd systemu

Efekt dydaktyczny

- Zrozumienie PK (Primary Key) i FK (Foreign Key)
- Zrozumienie relacji
- Zrozumienie pojęcia: integralność danych
- Zauważenie bazy danych jako systemu

Normalizacja
Problem złego projektu bazy danych

Cel: zrozumienie, jak projektować poprawne tabele i unikać błędów

- Jedna duża tabela:
  - powielanie danych
  - trudność aktualizacji
  - ryzyko błędów
  - brak spójności

Normalizacja
Problem złego projektu bazy danych

- Wyobraźmy sobie złą bazę danych.
- Jedna tabela zawiera:
  - pacjenta
  - lekarza
  - wizytę
  - receptę
  - lek
- Czyli coś takiego:

  | Pacjent | Lekarz | Data | Lek |
- Na pierwszy rzut oka wygląda to prosto…

Normalizacja
Problem złego projektu bazy danych

- Ale pojawiają się problemy:
  - dane pacjenta powtarzają się wiele razy
  - dane lekarza powtarzają się wiele razy
  - zmiana jednego pola wymaga wielu aktualizacji
- To prowadzi do:
  - redundancji
  - niespójności
  - błędów

Normalizacja
Problem złego projektu bazy danych

Jaki problem występuje przy jednej dużej tabeli?

A powielanie danych
B mniej danych
C łatwiejsze zarządzanie
D szybsze działanie

Czym jest normalizacja?

- **Normalizacja** to proces:
  - dzielenia danych na tabele
  - eliminowania redundancji
  - poprawy spójności

- Normalizacja to sposób projektowania bazy danych, który ma jeden cel:
  - **usunąć powielanie danych**

Czym jest normalizacja?

- Robimy to poprzez:
  - podział danych na osobne tabele
  - tworzenie relacji między nimi
- Czyli:
  - zamiast jednej tabeli → mamy wiele powiązanych tabel
- To dokładnie to, co zrobiliśmy:
  - PACJENT
  - LEKARZ
  - WIZYTA
  - RECEPTA
- To jest przykład poprawnej normalizacji.

Czym jest normalizacja?

Jaki jest główny cel normalizacji?

A zwiększenie liczby tabel
B przyspieszenie komputera
C zmniejszenie liczby użytkowników
D **eliminacja redundancji danych**

Podstawowe formy normalne

- Nie będziemy wchodzić w bardzo formalne definicje — skupimy się na intuicji.
  - **1NF (pierwsza postać normalna)** każda komórka zawiera jedną wartość (brak list w polach)
  - **2NF** wszystkie dane zależą od klucza
  - **3NF** brak zależności między atrybutami
- Najważniejsze do zapamiętania:
  - dane mają być podzielone logicznie i bez powielania

Podstawowe formy normalne

Co oznacza 1NF?

A brak relacji
B brak powtarzających się pól
C brak kluczy
D brak tabel

Efekt dydaktyczny

- Zrozumienie problemu złej struktury danych
- Zrozumienie sens normalizacji
- Zrozumienie podziału na tabele
- Przygotowanie na projektowanie zapytania

Zapytania QBE i SQL
Czym są zapytania?

- **Zapytanie (query)**
  - to sposób pobierania danych z bazy.
- Pozwala:
  - wyszukiwać dane
  - filtrować dane
  - sortować dane
  - analizować dane

Zapytania QBE i SQL
Czym są zapytania?

- Do tej pory stworzyliśmy:
  - tabele
  - relacje
- Ale baza danych bez zapytań jest praktycznie bezużyteczna.
  - zapytania pozwalają „zadawać pytania bazie danych”
  - nadają sens budowania bazy danych

Zapytania QBE i SQL
Czym są zapytania?

Przykłady:

- znajdź wszystkich pacjentów
- znajdź wizyty danego lekarza
- znajdź recepty z danego dnia

- Zapytania są podstawowym narzędziem pracy z bazą danych.

Zapytania QBE i SQL
Czym są zapytania?

Do czego służą zapytania?

A tworzenia tabel
B tworzenia raportów
C zarządzania systemem
D pobierania danych

QBE (Query By Example)

- QBE – zapytania graficzne (Access)
  - wybór tabel
  - wybór pól
  - ustawienie warunków
- Bez pisania SQL

QBE (Query By Example)

- QBE to podejście wizualne.
- W Accessie projektant:
  - wybiera tabelę
  - zaznacza pola
  - wpisuje warunki
- I system sam tworzy zapytanie.
- To świetne narzędzie na początek, bo:
  - pozwala zrozumieć logikę zapytań
  - bez znajomości składni SQL
- Ale:
  - QBE to tylko „wizualna wersja SQL”

QBE (Query By Example)

QBE pozwala tworzyć zapytania:

A tylko w SQL
B tylko w kodzie
C graficznie
D tylko w Excelu

Podstawy SQL

- SQL – język zapytań
- Podstawowe elementy:
  - SELECT
    FROM
    WHERE
- Każdy system baz danych używa SQL:
  - MySQL
  - PostgreSQL
  - SQL Server

Podstawy SQL

- SQL – język zapytań
- SQL to standardowy język pracy z bazami danych.
- Każde zapytanie składa się z trzech podstawowych części:
  - SELECT – co chcemy pobrać
  - FROM – z jakiej tabeli
  - WHERE – jakie warunki
- To jest absolutna podstawa.

Podstawy SQL

Które słowo w SQL określa, jakie dane pobieramy?

A FROM
B WHERE
C JOIN
D SELECT

Przykład zapytania SQL

```
SELECT ‚imię’, ‚nazwisko’
FROM ‚PACJENT’
WHERE miasto = 'Rzeszów';
```

Przykład zapytania SQL

- To jest pierwsze realne zapytanie SQL.
- Czytamy je tak:
  - wybierz imię i nazwisko
  - z tabeli PACJENT
  - gdzie miasto = Rzeszów
- Bardzo ważne:
  - SQL czytamy prawie jak język naturalny

Przykład zapytania SQL

Co robi klauzula WHERE?

A filtruje dane
B sortuje dane
C tworzy tabelę
D wybiera tabelę

Rodzaje zapytań

- Rodzaje zapytań:
  - wybierające (SELECT)
  - z warunkami (WHERE)
  - sortujące (ORDER BY)
  - agregujące (COUNT, SUM)

Rodzaje zapytań

- SQL pozwala robić znacznie więcej niż tylko pobierać dane.
- Możemy:
- filtrować dane (WHERE)
- sortować dane (ORDER BY)
- liczyć dane (COUNT)
- sumować dane (SUM)

Rodzaje zapytań

- Przykład:
  - ile wizyt miał lekarz
  - ilu pacjentów jest w systemie
- To jest moment, gdzie baza danych zaczyna być:
  - narzędziem analitycznym

Rodzaje zapytań

Która funkcja liczy liczbę rekordów?

A SUM
B COUNT
C MAX
D AVG

Efekt dydaktyczny

- Zrozumienie sensu zapytania
- Poznanie podstawy SQL
- Zrozumienie zastosowania i istoty QBE
- Zrozumienie analizy danych

Formularze, raporty, makra
Formularze (Forms)

- Cel: pokazanie jak użytkownik pracuje z bazą danych

- **Formularz**
  - wprowadzanie danych
  - edycja danych
  - wygodny interfejs użytkownika

Formularze (Forms)

- Do tej pory pracowaliśmy na tabelach.
- Ale:
  - użytkownik NIE pracuje bezpośrednio na tabelach
- Dlaczego?
  - są nieczytelne
  - łatwo popełnić błąd
  - brak kontroli danych

Formularze (Forms)

- Dlatego używamy:
  - **formularzy**
- Formularz pozwala:
  - wprowadzić nowego pacjenta
  - zapisać wizytę
  - dodać receptę
- To jest „okno aplikacji”.

Formularze (Forms)

Do czego służy formularz?

A tworzenia tabel
B wprowadzania danych
C generowania raportów
D tworzenia zapytań

Formularze (Forms)

- Formularz „Pacjent”:
  - imię
  - nazwisko
  - PESEL
  - telefon
- Przyciski:
  - dodaj
  - zapisz
  - usuń

Formularze (Forms)

- Wyobraźmy sobie formularz: „Dodaj pacjenta”
- Zamiast tabeli mamy:
  - pola tekstowe
  - przyciski
  - prosty układ
- Użytkownik:
  - wpisuje dane
  - klika „zapisz”
- System automatycznie zapisuje dane w tabeli.
- To ogromna różnica w użyteczności.

Formularze (Forms)

Gdzie trafiają dane z formularza?

A do raportu
B do zapytania
C do pliku tekstowego
D do tabeli

Raporty (Reports)

- **Raport**
  - prezentacja danych
  - zestawienia
  - wydruki
- Przykład:
  - lista wizyt

Raporty (Reports)

- Raporty służą do:
  - prezentacji danych
- Przykłady:
  - lista pacjentów
  - historia wizyt
  - raport dzienny lekarza

Raporty (Reports)

- Raport:
  - jest czytelny
  - może być drukowany (łatwiej go wydrukować)
  - ma strukturę dokumentu
- To jest coś, co trafia do:
  - Lekarza
  - administracji
  - kierownictwa

Raporty (Reports)

Do czego służy raport?

A edycji danych
B usuwania danych
C prezentacji danych
D wprowadzania danych

Makra (Macros)

- **Makra**
  - automatyzacja działań
  - wykonywanie operacji
  - reakcja na zdarzenia

Makra (Macros)

- Makra pozwalają:
  - automatyzować pracę

Przykłady:

  - po kliknięciu przycisku → zapisz dane
  - po otwarciu formularza → załaduj dane
  - po zapisie → pokaż komunikat

- Makra są bardzo ważne, bo:
  - pozwalają stworzyć „logikę aplikacji” bez programowania

Makra (Macros)

Do czego służą makra?

A przechowywania danych
B tworzenia tabel
C tworzenia raportów
D automatyzacji działań

Baza danych jako aplikacja

- System bazodanowy =
  - tabele
  - relacje
  - zapytania
  - formularze
  - raporty

Baza danych jako aplikacja

- To bardzo ważny moment:
  - baza danych to nie tylko tabele
- To cały system:
  - przechowywanie danych
  - przetwarzanie danych
  - prezentacja danych
- Czyli:
  - baza danych = aplikacja
- To dokładnie tak działają:
  - systemy bankowe
  - systemy medyczne
  - sklepy internetowe

Baza danych jako aplikacja

Co składa się na system bazodanowy?

A tylko tabele
B tylko formularze
C tylko raporty
D wiele elementów (tabele, formularze, raporty itd.)

Baza danych jako aplikacja

Co składa się na system bazodanowy?

A tylko tabele
B tylko formularze
C tylko raporty
D wiele elementów (tabele, formularze, raporty itd.)

Podsumowanie
Cały proces tworzenia systemu

- Proces: rzeczywistość
  → analiza systemu
  → model danych
  → baza danych
  → aplikacja
- To jeden z najważniejszych slajdów bloku dotyczącego baz danych bo pokazuje pełną ścieżkę:
  - mamy rzeczywistość (np. przychodnia)
  - analizujemy ją (Wykład 2)
  - tworzymy model danych
  - budujemy bazę danych (Wykład 3)
  - tworzymy aplikację (formularze, raporty)

Podsumowanie
Cały proces tworzenia systemu

Który etap następuje bezpośrednio po analizie systemu?

A testowanie
B implementacja
C instalacja
D model danych

Podsumowanie
Cały proces tworzenia systemu

- Blok trzech wykładów dotyczących bazy danych pozwolił:
  - tworzyć tabele
  - definiować klucze
  - budować relacje
  - stosować normalizację
  - tworzyć zapytania
  - rozumieć formularze i raporty

Podsumowanie
Cały proces tworzenia systemu

- Od strony praktycznej:
  - zaprojektować tabele
  - zrozumieć klucze
  - zrozumieć relacje
- Od strony użytkowej:
  - jak pobierać dane (SQL)
  - zrozumieć formularze i raporty
- To jest fundament do dalszych tematów:
  - zaawansowane SQL
  - optymalizacja
  - systemy baz danych