Bazy danych i systemy zarządzania bazami

Wykład VI

Wprowadzenie do analizy systemowej i projektowania.

Diagramy przepływu danych (DFD)
Słowniki danych (DD)
Diagramy związków encji (ERD)
Diagramy sieci przejść (STP)
Diagramy przepływu sterowania (CFD)
Zasady projektowania baz danych

Pojęcie i rodzaje komponentów systemu

Pojęcie systemu

System to *regularnie współpracująca lub współzależna grupa elementów tworzących jednolitą całość.*

Przykłady systemów:

- abstrakcyjne: liczbowy, algebraiczny, logiczny, filozoficzny;
- fizyczne: grawitacyjny, termodynamiczny, elektromagnetyczny;
- techniczne: energetyczny, telefoniczny, informatyczny;
- organizacyjne: kadrowo-płacowy, zarządzania, gospodarki materiałowej;
- biologiczne: nerwowy, trawienny, kostno-mięśniowy.

Systemy mogą być: naturalne vs. sztuczne, abstrakcyjne vs. materialne, informacyjne – energetyczne – materiałowe, autonomiczne – automatyczne – sterowane ręcznie.

Typowe komponenty systemu

- elementy składowe (pojęcie stanu),
- połączenia (pojęcie przepływu),
- wejścia i wyjścia, wyodrębnienie z otoczenia,
- model (sposób działania),
- cel działania i ocena jakości,
- ograniczenia i zakłócenia.

Systemy informatyczne

Typowe systemy informatyczne

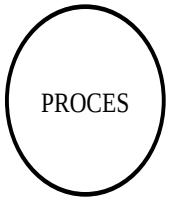
- systemy operacyjne,
- systemy obliczeniowe (obliczenia numeryczne i symboliczne),
- **systemy baz danych**,
- systemy z bazą wiedzy (AI, KBS, RBS, systemy eksperckie),
- systemy wspomagania decyzji, hurtownie danych,
- systemy graficzne, systemy wspomagania projektowania,
- systemy interakcyjne, systemy usługowe,
- systemy sterujące, systemy czasu rzeczywistego (*on-line*, *any-time*),
- systemy biurowe,
- systemy sieciowe.

Typowe elementy składowe systemów informatycznych

- sprzęt obliczeniowy (*hardware*),
- oprogramowanie (*software*),
- elementy sieci do przesyłu informacji,
- dane wejściowe i wyjściowe,
- algorytmy,
- ludzie.

Diagramy przepływu danych (DFD)

Elementy projektowania diagramów przepływu danych

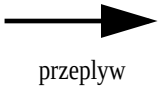


Procesy: reprezentacja funkcji systemu

Magazyny: reprezentacja zbiorow danych

Magazyny - inna reprezentacja

Terminatory: reprezentacja obiektow zewnetrznych



Przeplywy: reprezentacja przeplywu informacji

Podstawy konstrukcji diagramów przpływu danych (DFD)

Istota DFD

- DFD modelują funkcje systemu; są ukierunkowane na graficzne przedstawienie przepływów i transformacji danych,
- stanowią proste i intuicyjne narzędzie komunikacji projektanta i użytkownika,
- pozwalają na ograniczoną kontrolę poprawności i zupełności projektu.

Składniki DFD

- **Procesy**: definiują funkcje związane z przetwarzaniem informacji, realizowane są poprzez pewne procedury (algorytmy); posiadają *wejście* i *wyjście*;
- **Przepływy**: modelują przepływ informacji, są etykietowane rodzajem informacji, są ukierunkowane, mogą się schodzić i rozchodzić; przepływ informacji ma miejsce pomiędzy (i) procesami, (ii) terminatorami lub (iii) magazynami;
- **Magazyny**: modelują zbiory danych (zbiorniki, kontenery), stanowią elementy bierne, zachowują zawartość (odczyt nieniszczący);
- **Terminatory**: modelują obiekty zewnętrzne z którymi komunikuje się system, ich obecność sygnalizuje istnienie interfejsu; są odizolowane i niezależne od systemu.

Zasady konstrukcji diagramów przpływu danych (DFD)

Wytyczne dla konstrukcji DFD

Poprawnie skonstruowany diagram DFD powinien być logicznie (wewnętrznie) niesprzeczny i zgodny z zachowaniem modelowanego systemu.

Wytyczne dotyczące spójności wewnętrznej:

- wszystkie elementy diagramu powinny być etykietowane (możliwie jasno, prosto i jednoznacznie).
- nie modeluje się przepływów pomiędzy terminatorami (bezpośrednich),
- w diagramie nie powinny występować procesy bez wyjścia (tzw. nieskończone studnie),
- w diagramie nie powinny występować procesy bez wejścia (tzw. procesy spontanicznej generacji,
- w diagramie nie powinny występować magazyny tylko dla odczytu lub tylko dla zapisu (takie magazyny mogą być modelowane jako terminatory),
- każdy przepływ powinien mieć początek i koniec w odpowiednim elemencie grafu DFD (terminatorze, procesie lub magazynie); powinny to być różne elementy,
- w przypadku diagramów hierarchicznych należy zadbać o zgodność digramów wyższego i niższego poziomu (zgodność wejść i wyjść),
- diagram powinien być zrównoważony, tzn. być podzielony na odpowiednią liczbę poziomów, a każdy diagram powinien być o podobnym stopniu komplikacji.

Słowniki danych

Istota słownika danych

Słownik danych to wykaz wszystkich nazw danych (atrybutów) wraz z ich objaśnieniem (opisem), określeniem alfabetu, składni i/lub dziedziny oraz opisem semantyki.

Notacja słownika danych

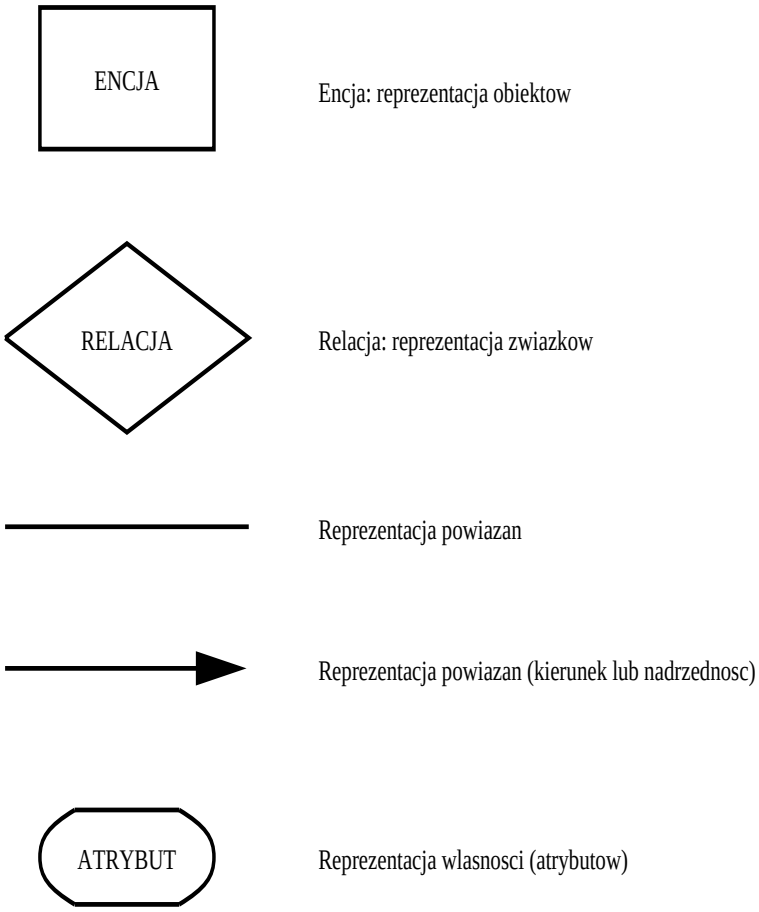
=	składa się z
+	i/oraz (konkatenacja)
()	element opcjonalny
{}	iteracja (powtórzenia)
[ ]	alternatywa (wybór jednej z możliwości)
|	rozdzielenie możliwości wyboru
-	znak zakresu
* *	komentarz
@	identyfikator klucza

Przykład:

identyfikator =	kod + numer
kod =	{Litera}
numer =	{cyfra}
Litera =	{A-Z}
cyfra =	{0-9}

Diagramy związków encji (ERD)

Elementy projektowania diagramów związków encji



Konstrukcja diagramów związków encji (ERD)

Komponenty diagramów ERD

- typy obiektów (klasy encji),
- charakterystyki obiektów (atrybuty encji),
- związki pomiędzy obiektami (związki encji, relacje),
- powiązania, powiązania skierowane (relacje niesymetryczne), powiązania hierachiczne, relacje etykietowane (1 do 1, 1 do N, N do N).

Wytyczne konstrukcji diagramów ERD

- relacje zachodzą pomiędzy obiektami,
- linie powiązania łączą obiekty poprzez relacje,
- atrybuty (o ile występują) przypisane są bezpośrednio do odpowiednich obiektów,
- relacje wiele-do-wielu są dekomponowane na dwa związki jeden-do-wielu,
- istotne relacje (posiadające atrybuty) są encjami (wprowadzanie encji pośrednich; czasem – abstrakcyjnych),
- należy rozróżniać encje i atrybuty (2NF, 3NF),
- encje powinny być elementarne (brak dekompozycji, podtypów) a atrybuty mieć wartości atomiczne (1NF, 4NF).

Projektowanie bazy danych – analiza DFD

Elementy bazy danych wynikające z analizy DFD

Analiza diagramów DFD pozwala określić (jakościowo) konieczność uwzględnienia następujących elementów:

- **zbiorniki danych** – tablice RBD; jeden zbiornik może być modelowany wieloma tablicami (opis w słowniku danych),
- **terminatory** – wskazują na konieczność budowy interfejsu (formularze, raporty, transfer danych),
- **procesy** – przetwarzanie danych (przetwarzanie transakcyjne – transakcja SQL; kwerenda, kwerenda funkcjonalna, makro; użycie funkcji i wyrażeń),
- **przepływy** – przesyłanie informacji; powiązanie elementu przetwarzania i składowania,
- **etykiety przepływów** – informują (jakościowo) o charakterze danych (opis w słowniku danych),
- **rozgałęzienia na wyjściu procesu** – operacje warunkowe lub rozbicie danych,
- **kombinacja przepływów wchodzących do procesu** – złączenie danych.

Wszystkie etykietowane elementy powinny zostać odzwierciedlone w strukturach lub elementach RBD (kompletność ptojektu).

Projektowanie bazy danych – analiza ERD

Struktura i elementy schematu tablicy

W wyniku *projektowania konceptualnego* otrzymuje się diagram ERD stanowiący abstrakcyjny model *struktury relacyjnej bazy danych*. W oparciu o diagram ERD wykonuje się projekt logiczny obejmujący strukturę i elementy tablic oraz powiązania między nimi. Projekt struktury logicznej obejmuje następujące czynności:

- **mapowanie encji** – mapowanie encji (silnych, słabych) na tablice RBD,
- **mapowanie encji asocjacyjnych** – mapowanie encji asocjacyjnych na tablice relacyjnej bazy danych opisujące atrybuty połączenia,
- **selekcja atrybutów** – wybór atrybutów; mapowanie atrybutów na kolumny tabel,
- **analiza wymagalności** – określenie wymagalności lub opcjonalności atrybutów,
- **analiza kluczy** – wybór kluczy głównych; określenie (identyfikacja) pozostałych kluczy,
- **mapowanie relacji** – mapowanie relacji na klucze obce,
- **analiza opcji nadtyp–podtyp** – wybór opcji dla relacji nadtyp–podtyp,
- **analiza opcji wykluczania** – wybór opcji dla relacji wzajemnego wykluczania się.

Projektowanie własności tablic

Techniczne definiowanie tablic: szczegóły

- określenie typu dla każdego atrybutu (przykłady),
- określenie rozmiaru każdego atrybutu,
- określenie wymagalności (opcjonalne vs. `NOT NULL`),
- ograniczenia lokalne (reguła poprawności, komunikat o błędzie),
- ograniczenia na poziomie rekordu (własności tablic),
- ograniczenia globalne, w tym:
- zdefiniowanie indeksów jednoznacznych (lub z powtórzeniami),
- zdefiniowanie ograniczeń globalnych,
- format danych,
- maska wprowadzania,
- wartość domyślna.

Dla SQL-92 określono następujące, zalecane opcje:

- `PRIMARY KEY` – oznacza kolumnę wybraną jako klucz główny,
- `UNIQUE` – gwarantuje jednoznaczność, ale dopuszcza (pojedynczą) wartość `NULL` w kolumnie,
- `DEFAULT` – domyślna wartość atrybutu, nadawana automatycznie,
- `CHECK` – definiuje dziedzinę atrybutu (nakłada **regułę poprawności**,
- `REFERENCES` oraz `FOREIGN KEY` – określają wiązanie klucza głównego z obcymi.

Mapowanie obiektów – zasady

Mapowanie obiektów prostych

Dla obiektów prostych (nie posiadających podtypów i nie będących nadtypami) zasady mapowania na tablice RBD są następujące:

- każdy obiekt jest mapowany na pojedynczą tablicę,
- każdy atrybut jest mapowany na kolumnę tej tablicy (z zachowaniem nazwy),
- unikalny identyfikator obiektu staje się kluczem głównym, lub
- składowe jednoznacznego identyfikatora tworzą klucz główny,
- zostają zachowane już określone własności atrybutów; można określić nowe własności.

Użytecznym standardem jest przyjęcie liczby mnogiej nazwy (typu, klasy) encji za nazwę tabeli (np. encja: KLIENT – tabela KLIENCI).

Mapowanie obiektów asocjacyjnych

Zasady mapowania na tablice RBD dla obiektów asocjacyjnych (relacji):

- każdy obiekt asocjacyjny jest mapowany na pojedynczą tablicę,
- każdy atrybut jest mapowany na kolumnę tej tablicy,
- unikalne identyfikatory łączonych obiektów stają się kluczami obcymi,
- suma atrybutów tworzących klucze obce tworzy klucz tabeli,
- dodatkowo, może zostać określony pojedynczy unikalny identyfikator łączonych obiektów; staje się kluczem głównym (klucz prosty),
- zostają zachowane już określone własności atrybutów.

Mapowanie relacji – zasady

Mapowanie relacji

Dla nazwanych relacji zasady mapowania na klucze obce są następujące:

- rozważmy relację 1–N; określamy tablicę nadrzędną (po stronie 1) i podrzędną (po stronie N); tabele te mają już określone atrybuty bazowe,
- w tabeli nadrzędnej wyszukujemy klucz główny i włączamy go jako klucz obcy do tabeli podrzędnej; identyfikuje on pojedynczy rekord tablicy nadrzędnej do którego dany rekord w tabeli podrzędnej jest przypisany,
- jeżeli istnieją dalsze tabele nadrzędne, to rekurencyjnie powtarzamy powyższą procedurę dobierając następne klucze obce dla tablic podrzędnych,
- dokonujemy ew. przemianowania kluczy obcych,
- sporządzamy dokumentację kluczy obcych.

Dalsze wskazówki dotyczące mapowania relacji – ACCESS:

- wymuszanie więzów integralności (standardowo),
- kaskadowa aktualizacja powiązanych pól (standardowo),
- kaskadowe usuwanie powiązanych rekordów (opcja),
- uwzględnienie typu złączenia (INNER, LEFT OUTER, RIGHT OUTER).

Mapowanie struktur

Struktury nadtyp–podtyp

Podtyp encji jest encją posiadającą własne (specyficzne) atrybuty, ale dziedziczącą atrybuty typu głównego (od szczytu hierarchii). W RBD istnieją trzy podstawowe możliwości modelowania logicznego strutur nadtyp–podtyp:

- **w pojedynczej tablicy** – zaprojektowanie oddzielnej (pojedynczej) tablicy zawierającej informację o nadtypie i wszystkich podtypach,
- **oddzielne tablice dla podtypów** – zaprojektowanie oddzielnych tablic (dla każdego podtypu),
- **oddzielne tablice dla nadtypu i podtypów** – interpretacja podtypów poprzez relację wzajemnego wykluczania się.

Mapowanie na pojedynczą tablicę przebiega następująco:

- do projketu tablicy wynikowej włączamy wszystkie atrybuty tablicy nadtypu,
- następnie włączamy wszystkie atrybuty podtypów ale z pominięciem unikalnych identyfikatorów,
- dołączamy kolumnę wskaźnika typu.

Cechy takiego mapowania to:

- bezpośredni dostęp do nadtypu (i jego atrybutów),
- możliwość dostępu do podtypów za pomocą widoków (perspektyw),
- możliwość istnienia atrybutów nie stosowalnych dla niektórych obiektów w jednej tabeli.

Mapowanie struktur – c.d.

Mapowanie na oddzielne tablice przebiega następująco:

- do każdej tablicy włączamy atrybuty nadtypu,
- do każdej tablicy włączamy atrybuty pojedynczego podtypu.

Cechy takiego mapowania to:

- prosta logika aplikacji,
- dostęp do nadtypu wymaga operacji UNION.

Mapowanie w sytuacji wzajemnego wykluczania się:

- wszystkie obiekty są mapowane na osobne tablice (osobno nadtyp i podtypy),
- relacja wykluczania jest reprezentowana jako klucz obcy.

Cechy takiego mapowania to:

- wszystkie obiekty są mapowane na osobne tablice (osobno nadtyp i podtypy),
- możliwość dodawania dalszych podtypów,
- dostęp do nadtypu wymaga operacji JOIN.

Zasady postępowania przy konstrukcji ERD i tablic

Dobre praktyki

Poniższe zasady stanowią przykłady tzw. *dobrych praktyk*:

- **encje $\neq$ atrybuty** – należy rozróżnić **encje** i **atrybuty**; decyduje o tym charakter danego obiektu i specyfika zastosowań (atrybut staje się encją gdy ma sam dalsze atrybuty),
- **schemat nazw tabel** – jednolity schemat nazw tabel, przejrzystość, krótkie nazwy, bez znaków specjalnych i liter narodowych,
- **jednoznaczna identyfikowalność encji** – należy zidentyfikować (lub określić) klucz (prosty lub złożony) dla umożliwienia jednoznacznej identyfikacji encji,
- **jednokrotne występowanie atrybutów** – atrybut występuje tylko raz, przy encji którą opisuje; powtarzające się atrybuty należy wyeliminować,
- **atomiczność atrybutów** – wartości atrybutów muszą być atomiczne; pojęcie atomiczności zależy od zastosowania,
- **implikacja generacji encji** – gdy istnieje powtarzający się atrybut, o podobnym charakterze (np. nr siedzenia),
- **jednolitość nazewnictwa** – nazwy encji i atrybutów powinny być jednolite i rozróżnialne (tablice – l. mnoga, atrybuty – l. pojedyncza),

Rozwiązywanie problemów

Problemy reprezentacji

- **reprezentacja obszernych identyfikatorów** – poprzez nadanie im kodów i uzupełnienie systemu o tablicę kodów oraz odpowiednie wiązanie,
- **reprezentacja stringów strukturalnych** – poprzez użycie pól segmentowych i funkcji dla ich rozkładu,
- **reprezentacja zbiorów, pola wielowartościowe** – w osobnej tabeli dwukolumnowej; nazwa zbioru jest wartością pierwszego atrybutu, a wyliczenie elementów następuje w drugiej kolumnie,
- **reprezentacja identyfikatorów** – użycie operatora `LIKE` z odpowiednim wzorcem danych,
- **reprezentacja przedziału wartości** – użycie operatora `BETWEEN ... AND ...` lub operatorów relacyjnych i koniunkcji,
- **reprezentacja porządku semantycznego** – powiązanie z tablicą definiującą ten porządek,
- **porównywanie wartości z 2,3, ... rekordów** – użycie odpowiedniego złączenia,
- **reprezentacja struktur** – rozbicie struktury na relacje (np. binarne) i reprezentacja w tabeli dwukolumnowej,
- **złożone przetwarzanie warunkowe** – symulacja systemu regułowego poprzez reprezentację reguł definiujących warianty przetwarzania w tabeli i odpowiednie zastosowanie θ-złączenia.

Rozwiązywanie problemów

Problemy ograniczeń

- **problemy z autonumerem** – przedefiniowanie typu na *liczba całkowita długa* i modyfikacja,
- **ograniczenie dziedziny do listy wartości** – zdefiniowanie listy wartości w regule poprawności,
- **ograniczenie dziedziny do zbioru** – zdefiniowanie encji nadrzędnej wraz z możliwymi wartościami oraz powiązania z wymuszeniem więzów integralności,
- **wymuszenie rozróżnialnych wartości w pojedynczej kolumnie** – zdefiniowanie indeksu jednoznacznego,
- **wymuszenie rozróżnialnych wartości w kilku kolumnach** – zdefiniowanie indeksu wielopolowego (złożnego, wielokrotnego),
- **ograniczenie liczby rekordów** – podwiązanie pola w relacji 1 – 1 do zadanej tabeli definiującej nazwy (wartości atrybutów) i liczbę rekordów,
- **reprezentacja wartości alternatywnych** – użycie funktora `IN` z listą wartości,
- **zbyt dużo atrybutów (>255)** – rozbicie na tablice poziome,
- **ograniczenia w SQL** – użycie opcji `PRIMARY KEY`, `UNIQUE`, `DEFAULT`, `CHECK`, `REFERENCES`, `FOREIGN KEY`.

Koniec

A mo£e poczśtek...