Projektowanie Baz Danych
Skrypt do wykładu prof. Nguyena

Paweł Abramowicz
http://abramowicz.org

24 stycznia 2015

Spis treści

Rozdział 1

Wstęp

1.1 Wstępne notatki u prof. Nguyena

W trakcie kursu *Projektowanie Baz Danych / Projekt* stworzymy dwa projekty bazy danych: projekt bazy *relacyjnej* i *obiektowej*.

Kurs *PBD / Wykład* kończy się kolokwium zaliczeniowym, w którym za ocenę z projektu zostaną doliczone dodatkowe punkty. Kolokwium poprawkowe zostanie zorganizowane w sesji[1].

Przelicznik punktowy oceny z projektu na dodatkowe punkty z kolokwium:

5.5 +4 pkt

5.0 +3 pkt

4.5 +2 pkt

4.0 +1 pkt

Większość materiałów do wykładu znajduje się na tzw. Boardzie[2] – loginem jest numer indeksu, hasło jest identyczne z hasłem do poczty studenckiej; w związku z likwidacją Instytutu Informatyki nie jestem w stanie przewidzieć, gdzie znajdą się materiały w roku 2015/16.

W obecnym kursie nie będziemy omawiać podstaw języka *SQL* – jeśli będzie on wykorzystywany, to zwykle z omówieniem; jeśli ukaże się mój skrypt do kursu *Bazy Danych*, zapewne włączę weń krótki kurs *SQL*a.

[1] Dane za semestr zimowy 2014/15

[2] `https://eportal.ii.pwr.edu.pl`

Kurs nie zawiera też opisu przygotowania interfejsu i raportów; przydałoby się to na kursie z Baz Danych, jak i tutaj, nie będzie jednak na kolokwium.

Rozdział 2

Podstawy Projektowania Baz Danych

Projektowanie bazy danych, jak każdego systemu informatycznego, zaczynamy od odpowiedzi na cztery pytania:

po co? – obieramy *cel* przedsięwzięcia

co? – definiujemy *zakres* projektu

jak? – określamy *sposób* wykonania systemu

czym? – wybieramy *narzędzia*.

W projekcie bazy danych będziemy wyróżniać trzy fazy:

1. Konceptualną. Do fazy konceptualnej należy analiza świata rzeczywistego i tworzenie diagramu obiektowo-związkowego (*O-Z*, *ERD*).
2. Logiczną. W fazie logicznej projektuje się schemat relacyjny, normalizuje się bazę i tworzy schemat logiczny.
3. Fizyczną. Definiuje się więzy integralności, projektuje raporty, interfejs, grupy użytkowników i planuje archiwizację[1, s. 1].

2.1 Podstawowe pojęcia

Dane dowolne wartości określonego typu

Informacje dana + przyporządkowanie jej do świata rzeczywistego

Wiedza informacja + mechanizm wnioskowania; ważne jest tutaj właśnie *podejmowanie decyzji*

Baza danych zbiór powiązanych ze sobą logicznie danych o określonych strukturach (tj. zorganizowany z określonym modelem danych [1, s. 1]).

System zarządzania bazą danych (DBMS) oprogramowanie bądź system informatyczny służący do zarządzania bazą danych [2]. Do zadań SZBD należą przetwarzanie danych (ich aktualizacja i wyszukiwanie) i zarządzanie więzami integralności (spójność danych); pełna lista:

- realizacja operacji dostępu do danych na poziomie fizycznym,
- zapewnienie integralności danych,
- obsługa współbieżności,
- ochrona danych,
- odtwarzanie po awariach,
- śledzenie operacji dokonywanych na bazie danych,
- obsługa rozproszenia,
- obsługa przetwarzania równoległego[1, s. 3].

System bazy danych System informatyczny umożliwiający dostęp do danych; SZBD wraz z interfejsem dla użytkownika końcowego.

Architektura systemu baz danych

poziom zewnętrzny/użytkownika – zbiór zewnętrznych widoków, przez które użytkownicy widzą zawartość bazy danych

poziom logiczny/modelu danych – zawiera model konceptualny bazy; model obejmuje całą zawartość bazy, tak jak widzi ją administrator lub właściciel bazy

poziom wewnętrzny/fizyczny – fizyczna reprezentacja bazy w postaci plików na dysku i algorytmów dostępu do nich.

Struktury/modele baz danych

- relacyjna
- obiektowa
- logiczna

Więzy integralności – zbiór reguł określających poprawne stany bazy danych [1, s. 1].

Transakcja – ciąg logicznie powiązanych operacji na danych. Umożliwiają spójne wykonanie złożonej operacji na danych, przez zastosowanie metody "wszystko albo nic". Cechy transakcji opisuje akronim *ACID*

Rozdział 3

Relacyjne Bazy Danych

3.1 Encje

Encja (ang. *entity*) to pojęcie bazowe, niemożliwe do zdefiniowania z formalnego punktu widzenia. Jest to reprezentacja wyobrażonego lub rzeczywistego obiektu lub ich grupy stosowana przy modelowaniu danych [5].

Encje z jednego **zbioru encji**, czyli grupy podobnych do siebie według określonych kryteriów encji, rozróżniamy na podstawie wartości atrybutów. Atrybuty posiadają następujące własności:

- ich liczba jest ustalona dla encji z jednego zbioru;
- ich typy są proste (wartości atrybutów są *atomowe*, albo niepodzielnie – tj. nie posiadają one struktury, z której można wyłonić mniejsze znaczące wartości);
- ich nazwy są jednoznaczne.

3.2 Klucze

Encja powinna posiadać klucz, czyli podzbiór atrybutów, który jest różny dla każdej z encji w zbiorze. Klucz ten powinien spełniać następujące własności:

- być jednoznaczny; nie mogą istnieć dwie encje w jednym zbiorze o identycznej wartości atrybutów kluczowych – wartość klucza ma jednoznacznie identyfikować encję.

– być minimalny (dot. kluczy złożonych); nie można usunąć żadnego atrybutu z klucza bez naruszenia wymogu jednoznaczności – każda składowa klucza jest potrzebna [1, s. 2].

Klucze mogą być **naturalne**, wynikające z rzeczywistej lub już stosowanej metody rozróżniania obiektów. Dla samochodów w Polsce może być to numer rejestracyjny, dla obywateli Polski PESEL; należy pamiętać, że jeśli wartość ta jest nieznana lub nie da się przyporządkować obiektowi wartości klucza naturalnego (np. obcokrajowiec nieposiadający numeru PESEL; pojazd, którego rejestracja jest niewymagana lub numery mogą kolidować z numerami rejestracyjnymi dla samochodów), nie powinno się stosować takiego klucza.

W takich przypadkach należy użyć **Klucza sztucznego**, czyli wartości arbitralnie nadanej przy wpisie do bazy, jednoznacznie identyfikującej encję w systemie. Przykładem może być numer faktury lub identyfikator użytkownika. Do niedawna identyfikatory użytkownika były stosowane dla celów autoryzacji i wyszukiwania (np. numer telefonu, numer w sieci *Gadu-Gadu* czy *ICQ*), jednak odchodzi się od takich rozwiązań na rzecz unikalnych *nazw użytkownika* (np. nazwa użytkownika w *Skype* dla potrzeb logowania i wyszukiwania, nazwa użytkownika na *Facebooku* do logowania; adres e-mail w wielu usługach).

Takie unikalne atrybuty lub grupy atrybutów nazywamy **kluczami kandydującymi**[1]. Może istnieć kilka kluczy kandydujących dla jednej bazy; czasem istnieją naturalne klucze kandydujące, jednak projektant decyduje się na użycie klucza sztucznego, np. ze względu na możliwość pojawienia się encji nieposiadającej wartości dla tego klucza (np. otworzenie punktu sprzedaży sklepu za granicą dla klucza kandydującego złożonego z numeru PESEL), albo ze względu na nadmiarowość informacji w tym kluczu (znów - klucz złożony z numeru PESEL może być nieoptymalny dla bazy dziesięciu osób, która ma wiele relacji odwołujących się do tychże osób przez klucze obce).

Klucz obcy, który pojawił się w ostatnim zdaniu, to nic innego, jak odwołanie do innej relacji – powiązanie między relacjami. Mając relacje: Klienci(NrKlienta, Nazwisko, Miasto) oraz Zamowienia(NrZamowienia, NrKlienta, Towar, Ilosc), kluczem obcym jest NrKlienta w relacji Zamowienia. Umiejscowienie klucza obcego w jednej z relacji powiązania lub oddzielnej relacji opisującej powiązanie jest zależne od liczności powiązania.

[1] Z formalnego punktu widzenia klucz kandydujący to atrybut lub zbiór atrybutów relacji, który spełnia warunki jednoznaczności i minimalności

3.3 Powiązania

Powiązania lub *związki* to opisy stosunków pomiędzy encjami w bazie. Do parametrów powiązania należą:

- Nazwa powiązania
- Rola encji w powiązaniu
- Opcjonalność (0..1, 0..n) / obligatoryjność (1..1, 1..n)
- Liczność (1–1, 1–n, n–n); do oznaczenia liczności możemy zamiast n użyć symbolu ∞
- Atrybuty powiązania (najczęściej w związkach wiele-do-wielu)

Przykładowe powiązanie n–n

Książka(Sygnatura)
| 0..n, jestWypożyczona
<>Wypożyczenie(dataWypoż, dataOdd, termin, kara, uwagi)
| 0..n, wypożycza
Czytelnik(PESEL)

3.3.1 Reprezentacja związków w relacjach

1–1 klucz obcy wstawiamy do dowolnej z relacji; można też rozważyć złączenie relacji, jeśli związek jest obligatoryjny.

1–n klucz obcy wstawiony do relacji po stronie n związku – czyli tej, która może mieć tylko jedno powiązanie z drugą relacją.

n–n stosujemy oddzielną relację o kluczu będącym złożeniem kluczy głównych związanych zbiorów encji (jeśli pary są unikalne) lub o kluczu sztucznym (jeśli zamierzamy przechowywać więcej niż jedną taką parę – np. spotkanie między dwoma osobami – ale lepiej pomyśl o wydzieleniu takiego związku w encję).

3.4 Normalizacja. Postaci normalne

Normalizacja to proces zapobiegający powtarzalności (redundancji) danych w bazie. Do zrozumienia wyższych postaci normalnych potrzebna jest definicja zależności funkcyjnych:

Atrybut Y zależy funkcyjnie od atrybutu X ($X \to Y$) wtedy i tylko wtedy, gdy w każdym momencie dla każdej wartości X istnieje dokładnie jedna wartość Y.

To znaczy, że jeśli istnieją w R dwie krotki z tą samą wartością X, to mają też tę samą wartość Y[1, s. 3]. Pamiętaj, że zależność ta nie działa w drugą stronę!

3.4.1 1PN. Pierwsza postać normalna

Atrybuty nie posiadają żadnej dodatkowej struktury (są atomowe, skalarne, typu prostego.

Znormalizowanie bazy do 1PN upraszcza i/lub przyspiesza porównywanie przez zmniejszenie kosztu samej operacji porównania.

3.4.2 2PN. Druga postać normalna

Każdy atrybut niekluczowy jest "w pełni" zależny funkcyjnie od klucza głównego. Ponadto relacja jest w pierwszej postaci normalnej.

Zależność "pełna" oznacza, że nie istnieje atrybut zależny funkcyjnie od części klucza (dla klucza $\{A, B\}$, nie istnieje zależność typu $A \to C$). Klucz jednoatrybutowy gwarantuje drugą postać normalną.

Postać druga zapobiega występowaniu anomalii (wprowadzania, usuwania, aktualizacji).

3.4.3 3PN. Trzecia postać normalna

Każdy atrybut niekluczowy jest nieprzechodnio zależny od klucza głównego. Ponadto relacja jest w drugiej postaci normalnej[2].

Zależność funkcyjna przechodnia zachodzi wtedy, gdy dla zależności funkcyjnej $X \to Y$ istnieje atrybut $Z(Z \neq Y, Z \neq X)$ taki, że $X \to Z$ i $Z \to Y$.

Postać trzecia w dalszym stopniu zapobiega redundancji danych i anomaliom.

[2] to oznacza, że jest także w pierwszej

3.4.4 Dobrze znormalizowana baza

Dobrze zaprojektowana relacja składa się z klucza głównego (prostego lub złożonego) i z pewnej liczby niezależnych od siebie atrybutów. Każdy atrybut zależy tylko od pełnego klucza głównego.

2PN dotyczy relacji ze złożonym kluczem głównym. Wymaga, by żaden atrybut niekluczowy nie zależał funkcyjnie od części klucza głównego. *3PN* wymaga, by każdy atrybut niekluczowy zależał tylko od klucza głównego[1, s. 3].

3.5 Więzy integralności

Więzy integralności to warunki nałożone na dane przechowywane w bazie. Określają one poprawne stany, w których może znajdować się baza.

Do więzów integralności bywają zaliczane następujące ujęcia [3]:

- **integralność relacji** (ang. *entity integrity*) związana jest z konceptem klucza głównego. Integralność relacji jest regułą, stanowiącą, że każda relacja musi mieć klucz główny – i że wartości atrybutów, które zostały zdefiniowane jako klucz główny powinny być unikalne i niepuste (ang. *not NULL*)
- **integralność odwołań** (ang. *referential integrity*) jest związana z kluczami obcymi. Integralność odwołań wymaga, by każda wartość klucza obcego była odwołaniem do istniejącej wartości klucza głównego innej relacji w bazie danych. W niektórych przypadkach, gdy encja w związku może mieć liczność 0, wartość klucza obcego może być pusta (być wartością *NULL*); w tym przypadku wartość *NULL* jest wyrażeniem niewystępowania lub braku informacji o danym związku pomiędzy modelowanymi obiektami.
- **integralność dziedziny** (ang. *domain integrity*) stanowi o tym, że wszystkie atrybuty muszą mieć zdefiniowaną dziedzinę. Dziedzina musi składać się z elementów niepodzielnych (atomowych).
- **więzy integralności użytkownika** (ang. *user-defined integrity*) to zbiór reguł zdefiniowanych przez użytkownika, których nie można zakwalifikować do powyższych ujęć.

Warunki podane na wykładzie można zakwalifikować do powyższych zagadnień:

- warunki wynikające ze struktury logicznej stanowią o integralności dziedziny i relacji
- warunki wynikające z rzeczywistości to elementy *user-defined integrity*; należą do nich maski, czyli dodatkowe warunki na dane: DataUr < Data - 18 lat, liczbaWypożyczeńStudenta <= 10 (and >= 0). Niektóre z warunków rzeczywistych zostają zapewnione już na poziomie struktury logicznej: imię to łańcuch znaków o zmiennej długości, maksymalnie 32 znaki; data urodzenia to data; id użytkownika jest autonumerowane.

3.5.1 Przykłady więzów integralności

DataWystawieniaFaktury >= DataSprzedaży

Jeśli klient.sumaZamowien() >= 10000 zł, to: TerminPlatnosci <= DataWystawienia + 30 dni;

Jeśli nie:

Jeśli KwotaFakturyNetto <= 10% * WartoscMagazynu, to:

TerminPlatnosci <= DataWystawienia + 14 dni

Jeśli nie: TerminPlatnosci = DataWystawienia.

Rozdział 4

Obiektowe Bazy Danych

Bazy obiektowe różnią się od relacyjnych strukturą danych; zamiast w tabelach (relacjach), dane przechowywane są w obiektach i powiązane referencjami. Wymusza to zmianę paradygmatu myślenia: w bazach obiektowych w polu można przechowywać nie tylko typy proste, ale także referencje to innych obiektów, a także kolekcje – związki jeden-do-wielu i wiele-do-wielu nie wymagają już osobnych tabel/obiektów; obiekty mogą mieć swoje metody.

4.1 Pojęcia

Podstawowymi bloczkami, z których budować będziemy projekt bazy obiektowej są:

+ klasa – typ obiektu, definiuje pola i metody,
+ obiekt – odpowiednik krotki w relacji,
+ atrybut – może być złożony, dziedziną mogą być klasy i kolekcje,
+ metoda – w relacyjnej jako dodatek były procedury, tutaj mamy hermetyzację, czyli metody są w jednej klasie z danymi, na których operują,
+ referencja – odpowiednik związku, ale może być w jedną stronę,
+ dziedziczenie – w relacyjnej traktowane jako związek 0..1–1..1.

Diagram obiektowo-związkowy (*ERD*) w bazie obiektowej budujemy, korzystając z *UML*a, standardu modelowania struktury klas w programowaniu obiektowym.

4.2 OQL

Językiem, za pomocą którego na wykładzie będziemy opisywać operacje na bazie obiektowej to **OQL** – *object query language*. Ważnym jego elementem jest **ścieżka**, czyli opis atrybutów i referencji potrzebnych do dotarcia do interesującego nas obiektu/atrybutu.

Podstawową klauzulą *OQL*a jest dobrze znane SELECT . . . FROM . . . WHERE.

Jako że dziedzina może być klasą lub kolekcją, ścieżka może składać się z więcej niż dwóch nazw (w relacyjnej: tabela.pole):

obiekt (czyli *relacja*) – *pole* (obiektowe) – . . . – *pole* (proste lub obiektowe)

Dużo częściej niż w relacyjnej będziemy używać odpowiednika instrukcji IN, który w *OQL*u ma postać IS-IN. Dzieje się tak z tego samego powodu, co zmiany w ścieżce: pola mogą być kolekcjami.

Rozdział 5

Transakcje

No dobrze, mamy zaprojektowany system, ale co dalej? Udostępniamy go użytkownikom, najlepiej jak największej ilości. Ale co jeśli dwóch użytkowników odwoła się naraz do tych samych danych?

Pojawia się masa problemów z przerywaniem **transakcji**, czyli logicznych ciągów powiązanych operacji na danych, które można sklasyfikować w trzech kategoriach:

utrata aktualizacji , ang. *lost update* – transakcja zapisuje dane, nie zważając na wcześniejszą ich zmianę; przykładowo A i B chcą dodać do tego samego konta odpowiednio 120zł i 80zł, razem 200zł. A, po czym B odczytują kwotę będącą na koncie (0zł). A dodaje do konta 120zł, zapisując dane. B dodaje do wcześniej pobranej wartości 80zł, nadpisując dane. W ten sposób na koncie znalazło się 80zł zamiast 200zł!

brudny odczyt , ang. *dirty read* – transakcja odczytuje dane związane z transakcją, która została zaniechana; przykładowo A i B to zlecenia przelewu po 100zł z dwóch różnych kont na jedno inne konto, razem 200zł. A odczytuje stan konta i dodaje do niego 100zł; dane zapisuje. B odczytuje nowy stan konta i dodaje do niego 100zł. W tym momencie okazuje się, że konto, z którego wysłano przelew A nie ma wystarczających środków do przelania 100zł, więc transakcja A zostaje zaniechana, konto wraca do wartości początkowej. B zapisuje stan konta po przelewie, uwzględniając jednak przelew A jako dokonany. Na koncie znajduje się 200zł, zamiast poprawnych 100zł.

niepowtarzalny odczyt , ang. *fuzzy read* – inna transakcja zmienia wartość danej między dwoma odczytami w jednej transakcji (drugi odczyt może służyć do potwierdzenia, że transakcja została wykonana poprawnie).

fantomy , ang. *phantom read* – inna transakcja dodaje do zbioru nową daną przed drugim odczytem w trakcie jednej transakcji; z punktu widzenia transakcji wygląda to jakby jej wykonanie miało skutek uboczny w postaci dodania nowej krotki.[6]

Rozwiązaniem powyższych problemów jest stosowanie transakcji o cechach *ACID*

A atomowość (ang. *atomicity*) – każda transakcja wykona się w całości, albo w ogóle;

C spójność (*consistency*) – po wykonaniu transakcji baza będzie spójna, tzn. nie zostaną naruszone żadne więzy integralności;

I izolacja (*isolation*) – *w skrócie* transakcje nie korzystają z danych edytowanych przez inną transakcję;

D trwałość (*durability*) dane wykonanej transakcji zostaną zapisane lub odtworzone z dziennika w razie awarii [4].

Dziennik transakcji (*transaction log*) to dokument zapisywany w pamięci trwałej, który zawiera przebieg wszystkich transakcji.

Transakcje mogą być zakończone w dwojaki sposób, zgodnie z zasadą atomowości:

COMMIT zatwierdza transakcję, nowe wartości zostają przeniesione do pamięci trwałej.

ROLLBACK nie przenosi do pamięci trwałej bazy żadnych danych; po transakcji nie zostaje żaden trwały ślad poza dziennikiem transakcji.

W razie awarii, system odtwarza dane z dziennika transakcji; służy do tego **mechanizm punktu kontrolnego**: w dzienniku wyznaczane są punkty czasowe, w których wiadomo, że baza była spójna i zostały wykonane wszystkie transakcje zatwierdzone przed tymże punktem. Ostatnio zatwierdzone transakcje znajdują się więc między przedostatnim, a ostatnim punktem kontrolnym.

Podczas przywracania systemu bazy danych tworzone są dwie listy:

REDO – transakcje do powtórzenia. Składa się z transakcji, które skończyły się sukcesem po ostatnim punkcie kontrolnym, ale przed awarią.

UNDO – transakcje do zaniechania. Transakcje zostały przerwane przez awarię w trakcie wykonywania.

Rozdział 6

Bazy rozproszone

6.1 Podstawowe pojęcia

Fragmentaryzacja danych – w celu rozproszenia zawartości bazy danych musimy w racjonalny sposób podzielić dane. Możemy stosować fragmentaryzację poziomą lub pionową:

Fragment poziomy Fragment składający się z wydzielonych krotek. Predykat, który wyznacza granice między fragmentami poziomymi nazywamy **dozorem**.

$$R = R_1 \cup R_2 \cup \ldots \cup R_n \tag{6.1}$$
$$R_i = \sigma_{w_i}(R) \tag{6.2}$$

W podanym wzorze w_i to właśnie dozór.

Fragment pionowy Fragment składający się z wydzielonych kolumn (wartości poszczególnych atrybutów).

$$S = S_1 \cup S_2 \cup \ldots \cup S_n \tag{6.3}$$
$$S_i = \pi_{A_i}(S) \tag{6.4}$$

W podanym wzorze A_i to zbiór atrybutów, wyznaczający fragment pionowy.

Dozór Predykat, czyli wzór, który sprawdza, czy krotka należy do danego fragmentu poziomego. Na przykład dla dwuserwerowej bazy danych studentów dozory mogą przyjmować następujące postaci:

$$w_1 = (NrAlbumu < 180000) \tag{6.5}$$
$$w_2 = (NrAlbumu \geq 180000) \tag{6.6}$$

Dane studenta o numerze indeksu 179635 byłyby zapisywane na serwerze 1, a dane autora tekstu, indeks numer 203417, byłyby przechowywane na serwerze 2.

6.2 Proces przetwarzania zapytań

Rozproszenie bazy danych umożliwia nie tylko większą pojemność, ale także zoptymalizowanie części poszukiwań [1]. W tym celu analizujemy **zapytanie wejściowe** i na jego podstawie **generujemy przestrzeń poszukiwań**. Za pomocą **reguł przekształceń**, które biorą pod uwagę fragmentaryzację poziomą i pionową, początkowe zapytanie jest przekształcane na równoważny *plan wykonania zadania*. Po sprawdzeniu kosztu zapytania (za pomocą **modelu kosztu**), na podstawie **strategii wyszukiwania** podejmowana jest decyzja, czy należy szukać dalszego rozwiązania (wracamy do przestrzeni poszukiwań), czy rozwiązanie jest optymalne (lub wystarczające) wg strategii.

Generowanie przestrzeni poszukiwań wyznacza wszystkie możliwe plany wykonania danego zapytania; każdy plan jest reprezentowany przez **drzewo** operacji, jakie trzeba wykonać, by uzyskać wynik.

Drzewa mogą mieć różne kształty, np. **liniowe** lub **krzaczaste**.

Reguły przekształcenia pozwalają wyznaczyć nowy plan wykonania, który jest równoważny poprzedniemu.

$$Kurs \triangleright\triangleleft (KS \triangleright\triangleleft Studenci) \Leftrightarrow Kurs \triangleright\triangleleft (KS \triangleright\triangleleft Studenci) \tag{6.7}$$
$$\sigma_{Indeks=123456}(Studenci \triangleright\triangleleft KS) \Leftrightarrow (\sigma_{Indeks=123456} Studenci) \triangleright\triangleleft KS \tag{6.8}$$

[1] Oczywiście wyszukiwanie nie będzie bardziej optymalne niż w przypadku nierozproszonej bazy, jednak może być bardziej optymalne od bazy zlokalizowanej na kilku dyskach sieciowych.

Rozdział 7

Kolokwium – Przykładowe pytania

Przykładowe pytania, mogące pojawić się na kolokwium. Źródło: slajdy prof. Nguyena na wykładzie 15.12.2014.

Zadania oznaczone *Nysa:* zostały pobrane z materiałów i notatek z przedmiotu Bazy Danych studentów informatyki w medycynie na PWSZ w Nysie[1]; wykład był prawdopodobnie prowadzony przez prof. Nguyena.

1. Ścieżką w języku OQL może być:
   (a) sekwencja zmienna-zmienna-. . . -zmienna
   (b) **sekwencja zmienna-atrybut-. . . -atrybut**
   (c) **sekwencja atrybut-atrybut-. . . -atrybut**
   (d) sekwencja klasa-klasa-. . . -klasa
2. Mechanizm punktu kontrolnego (*checkpoint*) w zarządzaniu transakcjami polega na: ([1, pytanie 1])
   (a) **wyznaczeniu co jakiś czas punktu czasowego w dzienniku i ostatecznym zatwierdzeniu transakcji zatwierdzonych między nim a ostatnim punktem kontrolnym.**
   (b) wyznaczeniu co jakiś czas punktu czasowego w dzienniku i zaniechaniu transakcji zatwierdzonych po nim

(c) wyznaczeniu co jakiś czas punktu czasowego w dzienniku i odtworzeniu transakcji zatwierdzonych po nim

(d) wyznaczeniu co jakiś czas punktu czasowego w dzienniku w celu lokalizacji punktu czasowego, w którym system ulega awarii

3. Lista REDO tworzona po awarii składa się z: ([1, pytanie 2])

   (a) **transakcji, które zostały zakończone pomyślnie przed awarią, ale po ostatnim punkcie kontrolnym**

   (b) transakcji, które zostały zakończone pomyślnie przed ostatnim punktem kontrolnym

   (c) transakcji, które były w trakcie realizacji w czasie awarii

   (d) transakcji, które zostały zaniechane przed awarią ale po ostatnim punkcie kontrolnym

4. Lista UNDO tworzona po awarii składa się z: (odpowiedzi poza ekranem) ([1, pytanie 3])

   (a) transakcji, które zostały zakończone pomyślnie przed awarią, ale po ostatnim punkcie kontrolnym

   (b) transakcji, które zostały zakończone pomyślnie przed ostatnim punktem kontrolnym

   (c) **transakcji, które były w trakcie realizacji w czasie awarii**

   (d) transakcji, które zostały zaniechane przed awarią ale po ostatnim punkcie kontrolnym

5. Polecenie COMMIT zleca: ([1, pytanie 9])

   (a) **zatwierdzenie transakcji, dokonane zmiany są zapamiętane na stałe**

   (b) zatwierdzenie transakcji, dokonane zmiany nie muszą być zapamiętane na stałe

   (c) natychmiastowe przerwanie transakcji

   (d) rozpoczęcie transakcji

6. Polecenie ROLLBACK zleca: ([1, pytanie 6])

   (a) zatwierdzenie transakcji, dokonane zmiany są zapamiętane na stałe

(b) rozpoczęcie danej transakcji

(c) natychmiastowe zakończenie transakcji

(d) **wycofanie wszystkich poczynionych od początku danej transakcji zmian**

7. ... między encjami to: (poza ekranem)

   (a) . . . (nieczytelne / poza ekranem)

   (b) atrybuty powiązania

   (c) opcjonalność powiązania (część poza ekranem?)

   (d) przeznaczenie powiązania

8. *Brudny odczyt* w transakcjach oznacza:

   (a) . . . ją sekwencyjny dostęp do danych (poza ekranem)

   (b) . . . ją jednoczesny dostęp do tych samych danych, z tym, że jedna z nich odczytuje . . . danych (poza ekranem)

   (c) . . . ją jednoczesny dostęp do tych samych danych, z tym, że jedna z nich odczytuje . . . na tych danych (poza ekranem)

   (d) . . . ją jednoczesny dostęp do tych samych danych, z tym, że obydwie chcą . . . ch danych (poza ekranem)

9. Proces projektowania relacyjnej bazy danych zawiera:

   (a) **fazę konceptualną**

   (b) **fazę logiczną**

   (c) fazę implementacji[1]

   (d) fazę analityczną

10. Faza konceptualna projektowania relacyjnej bazy danych zawiera:

    (a) **tworzenie diagramu obiektowo-związkowego (schemat konceptualny)**

    (b) **ustalanie dziedziny atrybutów**

    (c) ustalenie klucza głównego

[1] Mhm, istnieje coś takiego jak faza fizyczna, ale to co innego

11. Faza logiczna projektowania relacyjnej bazy danych zawiera:
    (a) **ustalenie klucza głównego dla każdego schematu**
    (b) ustalenie związków . . . (nieczytelne)
    (c) ustalenie raportów
    (d) **normalizację schematu**
12. Związek wiele do wielu może występować w:
    (a) **schemacie logicznym bazy relacyjnej**
    (b) schemacie . . . (nieczytelne)
    (c) schematach relacyjnych
    (d) więzach integralności (?)
13. Transakcja w bazach danych jest to: ([1, pytanie 4])
    (a) **ciąg logicznie ze sobą powiązanych operacji na danych**
    (b) . . . (nieczytelne)
    (c) jednostka pamięci
    (d) **sekwencja dowolnych operacji przeprowadzająca bazę danych z jednego spójnego stanu do drugiego**[2]
14. Właściwość *atomowość* transakcji oznacza: ([1, pytanie 6])
    (a) **wszystkie operacje należące do transakcji muszą być wykonane lub żadna z nich**
    (b) każda operacja musi być wykonana w całości
    (c) stan bazy danych przed i po wykonaniu transakcji musi być spójny
    (d) w tym samym czasie dwie transakcje nie mogą korzystać z tego samego obiektu danych
15. Właściwość *izolacja* transakcji oznacza: ([1, pytanie 7])
    (a) wszystkie operacje należące do transakcji muszą być wykonane lub żadna z nich
    (b) każda operacja musi być wykonana w całości

[2] dodałbym jeszcze operacji *na danych*

(c) stan bazy danych przed i po wykonaniu transakcji musi być spójny

(d) **w tym samym czasie dwie transakcje nie mogą korzystać z tego samego obiektu danych**

16. Powiązanie między klasami w bazach obiektowych to: ([1, pytanie 23])

(a) identyczność nazw klas

(b) **jedna klasa jest dziedziną atrybutu drugiej klasy (związek referencji)**

(c) przeciążenie metod

(d) enkapsulacja

(e) ich wspólna część nie jest pusta

(f) **dziedziczenie**

17. Skojarzenie klas w języku OQL odbywa się dzięki

(a) operacji złączenia klas

(b) *związkowi referencji między klasami* [3]

(c) operacji półzłączenia

(d) ścieżkom

18. W języku OQL zmienna może być:

(a) **typu logicznego**

(b) **typu referencyjnego**

(c) typu dziedzinowego[4]

(d) **związana przez predykat przynależności**[5]

19. Tabelę o schemacie $(\{A, B, C, D\}, \{A \to B, C \to D\})$ można podzielić na dwa fragmenty pionowe o atrybutach:

(a) $(\{A, B\}|\{C, D\})$

(b) $(\{A, B\}|\{A, C, D\})$ **OK**

[3] Skojarzenie klas to termin matematyczny i nie wiem jak go odnieść tutaj. Wydaje mi się, że to poprawna odpowiedź.

[4] D'uh, albo źle przetłumaczone, albo chodzi o to, żeby brzmiało źle, by odstraszyć nauczonych od zaznaczania

[5] Wydaje mi się, że chodzi o predykaty typu IS-IN, IS-SUBSET

(c) $(\{C, D\}|\{A, B, C\})$ **OK**

(d) $(\{A\}|\{B, C, D\})$

20. Dziennik transakcji służy do: ([1, pytanie 24])

(a) **zapisania przebiegu transakcji**

(b) przechowania aplikacji

(c) przechowania więzów integralnościowych

(d) zapisania słowników

21. Schemat $R = (U, F)$ jest w 2PN jeśli jest w 1PN oraz:

(a) **wszystkie atrybuty niekluczowe są w pełni zależne od klucza**

(b) wszystkie atrybuty niekluczowe są zależne od klucza

(c) nie ma atrybutów kluczowych

(d) zbiór U jest pusty

22. Schemat $R = (U, F)$ jest w 3PN jeśli jest w 1PN, 2PN oraz:

(a) **nie ma atrybutów niekluczowych**[6]

(b) **nie ma tranzytywnych**[7] **zależności atrybutów niekluczowych od klucza**

(c) zbiór F nie jest pusty

23. Właściwość *trwałość* transakcji oznacza: ([1, pytanie 8])

(a) wszystkie operacje należące do transakcji muszą być wykonane lub żadna z nich

(b) **wyniki działania transakcji muszą być zapisane do pamięci trwałej**

(c) stan bazy danych przed i po wykonaniu transakcji musi być spójny

(d) w tym samym czasie dwie transakcje nie mogą korzystać z tego samego obiektu danych

24. Pytanie 24 (poza ekranem)

[6] Jeśli nie ma atrybutów niekluczowych, to nie ma przechodnich zależności z odp. b ;)

[7] przechodnich – przyp. PA

25. Pytanie 25 (poza ekranem)
26. Pytanie 26 (poza ekranem)
27. Pytanie 27 (poza ekranem)
28. Pytanie 28 (poza ekranem)
29. Dla podanego schematu bazy danych, zapytanie *Podaj pracowników – właścicieli samochodów produkowanych przez firmy, w których pracują* w języku OQL ma postać[8]
    (a) SELECT C FROM Pracownik:C WHERE C-WłaścicielPojazdu-Producent = C-MiejscePracy (?)
    (b) SELECT C FROM Pracownik:C WHERE C-WłaścicielPojazdu-Producent = C-MiejscePracy-Nazwa (?)
    (c) SELECT C FROM Pracownik:C WHERE C-MiejscePracy-Nazwa = C-WłaścicielPojazdu (?)
    (d) SELECT C FROM Pracownik:C WHERE C IS-IN C-WłaścicielPojazdu-Producent-Pracownicy
30. Nysa: Optymalizacja zapytań na poziomie organizacji danych polega na: ([1, pytanie 15])
    (a) wyborze najmniej czasochłonnej procedury wykonania zapytań z uwzględnieniem konfiguracji komputera
    (b) optymalnej implementacji zapytania w wewnętrznym języku komputera
    (c) ...
31. Nysa: Optymalizacja zapytań na poziomie algebraicznym polega wyłącznie na: ([1, pytanie 16])
    (a) przekształcaniu zapytania do postaci równoważnej, w której liczba złączeń jest minimalna oraz koszt wszystkich operacji jest minimalny
    (b) budowie matrycy dla danego zapytania
    (c) przekształceniu zapytania do równoważnej postaci w rachunku krotkowym

[8] Odpowiedzi nie znam, bo nie mam obrazka ani pewności, że jest to dobrze przepisane.

(d) zapisywaniu zapytania w języku *SQL*

32. Nysa: Jedna ze strategii optymalizacji pytań relacyjnych wymaga, by: ([1, pytanie 17])

    (a) selekcję wykonać tak wcześnie, jak tylko możliwe

    (b) selekcję wykonać tak późno, jak tylko możliwe

    (c) projekcję wykonać przed złączeniem, kiedy tylko możliwe

    (d) w ogóle nie wykonać złączeń

33. Nysa: W podstawowym modelu obiektowym obiekt jest definiowany jako: ([1, pytanie 18])

    (a) **para <identyfikator, wartość>**

    (b) wartość pewnego typu złożonego

    (c) złożona wartość

    (d) zbiór procedur

34. Nysa: Enkapsulacja (hermetyzacja) jest to założenie o następującej treści: ([1, pytanie 19])

    (a) *obiekty danej klasy komunikują się z innymi obiektami tylko poprzez metody* [9]

    (b) jedna klasa może być typem wartości atrybutu innej klasy

    (c) klasa jest obiektem

    (d) identyfikator obiektu może być wartością atrybutu

35. Nysa: Właściwością transakcji jest: ([1, pytanie 5])

    (a) **atomowość**

    (b) **spójność**

    (c) dynamiczność

    (d) natychmiastowość

36. Nysa: Język relacyjny nazywamy językiem algebraicznym, jeśli: ([1, pytanie 11])

[9] Wydaje się być prawidłowe

(a) **oparty jest na algebrze relacyjnej**
(b) zawiera funkcje agregujące
(c) zawiera klauzulę SELECT FROM WHERE
(d) ma nazwę Query By Example

37. Nysa: Język relacyjny nazywamy zupełnym, jeśli: ([1, pytanie 13])

(a) może wyrażać dokładnie tyle samo, co algebra relacyjna
(b) może wyrażać co najmniej tyle samo, co rachunek kwantyfikatorów
(c) może wyrażać co najmniej tyle samo, co algebra relacyjna
(d) może wyrażać dokładnie tyle samo, co rachunek kwantyfikatorów

38. Nysa: Funkcje systemu zarządzania bazą danych to m.in.: ([1, pytanie 14])

(a) dbanie o spójność danych
(b) zarządzanie zapytaniami
(c) zarządzanie transakcjami
(d) takie same funkcje jak systemu operacyjnego

39. Nysa: Dziedziczenie klasy A od klasy B oznacza jednoznacznie, że: ([1, pytanie 20])

(a) klasa A zawiera te same obiekty co klasa B
(b) klasa A zawiera co najmniej ...
(c) ...

40. Nysa: (nieczytelne): ([1, pytanie 21])

(a) ...
(b) ...
(c) ... związku dziedziczenia
(d) możliwości wywołania różnych metod do danego obiektu

41. Nysa: Identyfikatorem obiektu: ([1, pytanie 22])

(a) powinna być wartość atrybutu kluczowego
(b) nie powinna być wartość atrybutu kluczowego
(c) powinna być niezależna wartość zadana przez system

Dzięki Pawłowi Kotasowi za uzupełnienie i weryfikację pytań oraz wskazanie strony informatyki w medycynie na PWSZ w Nysie.

Rozdział 8

Posłowie

No, to chyba wszystko co wiem na temat zbliżającego się kolosa z PBD, mam nadzieję, że skrypt był przydatny :)

Tutaj wkleję jeszcze trochę śmiecia, jakiego znalazłem.

1. W modelach obiektowych obiekt jest definiowany jako
   (a) dwójka (identyfikator, wartość)
   (b) instancja (element klasy)
   (c) wartość pewnego typu złożonego
   (d) pewien zbiór procedur
2. W modelu obiektowym klasa jest traktowana jako:
   (a) dowolny zbiór obiektów
   (b) obiekt innej klasy
   (c) zbiór obiektów o tym samym zbiorze atrybutów i metod
   (d) pewien typ wartości
3. Hermetyzacja jest to założenie o następującej treści:
   (a) obiekty danej lasy komunikują się z innymi obiektami tylko rzez metody
   (b) jedna klasa nie może być typem wartości atrybutu innej klasy
   (c) klasa jest obiektem

(d) identyfikator obiektu może być wartością atrybutu

4. Dziedziczenie klasy A od B oznacza:

   (a) klasa A zawiera te same atrybuty co B

   (b) klasa A zawiera co najmniej te same atrybuty co B

   (c) klasa A zawiera identyczne metody co klasa B

   (d) obiekty klasy B są dokładniej opisane niż obiekty klasy A

5. Mechanizm przeciążenia metod polega na:

   (a) używaniu tej samej nazwy dla metod tej samej klasy

   (b) używaniu tej samej nazwy i metod tej samej klasy

   (c) używaniu tej samej nazwy dla 2 metod należących do 2 różnych las nie będących w związku dziedziczenia

   (d) możliwości wywołania różnych metod do danego obiektu

Bibliografia

[1] http://www.infomed.cba.pl/ » Bazy Danych, Strona studentów informatyki w medycynie na PWSZ w Nysie. 2006

[2] http://pl.wikipedia.org/wiki/System_zarz%C4%85dzania_baz%C4%85_danych
Wikipedia: System zarządzania bazą danych
Historia edycji i autorzy: http://pl.wikipedia.org/w/index.php?title=System_zarz%C4%85dzania_baz%C4%85_danych&action=history

[3] http://en.wikipedia.org/wiki/Data_integrity
Wikipedia w jęz. ang.: Data integrity
Historia edycji i autorzy: http://en.wikipedia.org/w/index.php?title=Data_integrity&action=history

[4] http://en.wikipedia.org/wiki/ACID
http://pl.wikipedia.org/wiki/ACID
Wikipedia: ACID
Historia edycji i autorzy: http://en.wikipedia.org/w/index.php?title=ACID&action=history, http://pl.wikipedia.org/w/index.php?title=ACID&action=history

[5] http://pl.wikipedia.org/wiki/Encja
Wikipedia: Encja
Historia edycji i autorzy: http://pl.wikipedia.org/w/index.php?title=Encja&action=history

[6] M. Krótkiewicz, *Bazy danych*: Transakcje. Uniwersytet Opolski
http://www.math.uni.opole.pl/~mkrotki/bd_transakcje.pdf