PODSTAWY BAZ DANYCH – WPROWADZENIE

Baza danych to kolekcja powiązanych ze sobą danych, odpowiednio zorganizowana w celu szybkiego przeszukiwania i dostępu do informacji

- Charakterystyka bazy danych:
  - trwałość danych
  - rozmiar danych
  - złożoność danych
  - zintegrowanie

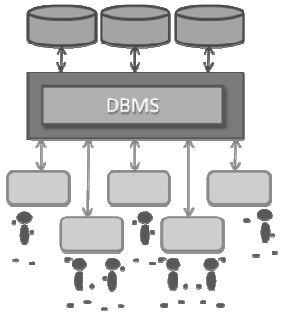
System zarządzania bazą danych (DBMS) to zestaw programów ogólnego przeznaczenia umożliwiający zarządzanie bazami danych i danymi przechowywanymi w bazie danych

- Podstawowe cechy i zadania DBMS
  - ochrona danych
  - integralność danych (tj. utrzymanie poprawności i aktualności)
  - wielodostępność
  - niezależność danych
  - integracja danych

System BD to skomputeryzowany system przechowywania i przetwarzania danych

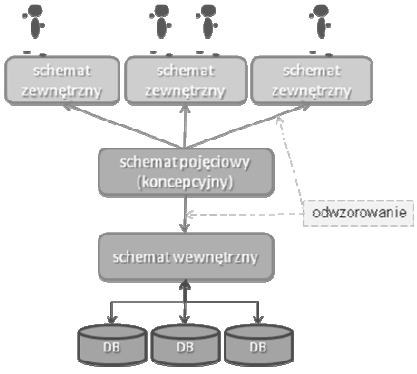
- Składniki systemu BD
  - dane (baza danych)
  - oprogramowanie (system hosta i DBMS)

  Użytkownicy (typowi: administrator BD, programista, użytkownik „naiwny")



Architektura systemu BD (standard ANSI/SPARC)

- poziom zewnętrzny
  - opisuje strukturę bazy danych z punktu widzenia określonej grupy użytkowników
- poziom pojęciowy
  - opisuje logiczną strukturę bazy danych
- poziom wewnętrzny
  - opisuje fizyczną organizację bazy danych: rozmieszczenie i organizację danych



- Niezależność danych
  - logiczna – możliwość zmiany schematu pojęciowego bez konieczności zmiany schematu zewnętrznego lub zmiany aplikacji
  - fizyczna – możliwość zmiany schematu wewnętrznego bez konieczności zmiany schematów pojęciowego i zewnętrznego

PODSTAWY BAZ DANYCH – PROJEKTOWANIE BAZY DANYCH

Kolejność projektowania

- zgromadzenie wymagań (m.in. analiza rzeczywistości i zidentyfikowanie danych)
- utworzenie modelu pojęciowego (np. ERD) (m.in. zachowanie zgodności z wymaganiami i potrzebami
- utworzenie schematu bazy danych (m.in. m.in. normalizacja)

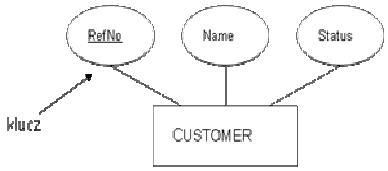
Model związków encji (ERD)

- model pojęciowy reprezentujący obiekty (encje) i zależności ze świata rzeczywistego

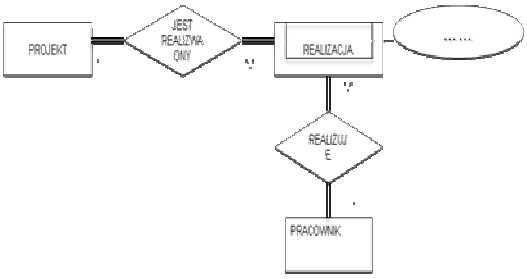
Podstawowe składniki

- encja (zbiór encji)
- atrybut (m.in. klucz)
- związek charakteryzowany przez: stopień (binarny - na 2 relacjach, unarny – na 1 relacji, ternarny – na 3 relacjach), typ (1:1, 1:N, N:M), obowiązkowość/opcjonalność

notacja Chena



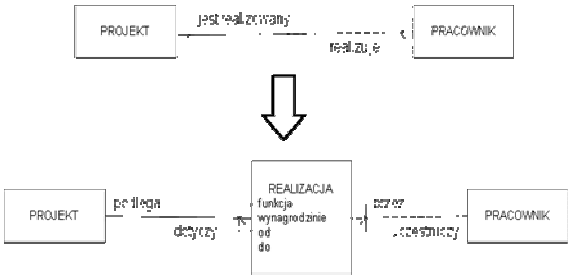
1:N, obowiązkowy od strony CAR →



N:M, obowiązkowy po stronie PROJEKT →
encja słaba: reprezentuje związek, może posiadać swoje własne atrybuty

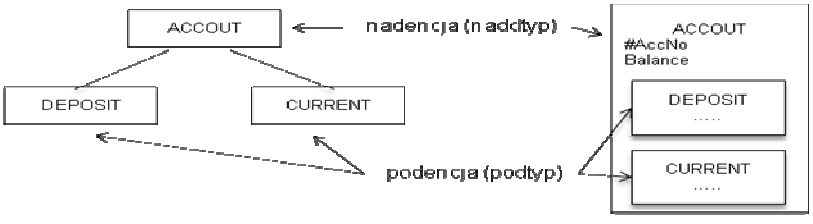
notcja Oracle'a

każda osoba może mieć wiele samochodów
jeden samochód należy do jednej osoby



każdy projekt jest realizowany przez wielu pracowników
każdy pracownik może uczestniczyć w wielu projektach

Związki generalizacji (specyfikacja podtypów)



PODSTAWY BAZ DANYCH – RELACYJNA BAZA DANYCH

Relacja to dowolny podzbiór iloczynu kartezjańskiego jednej lub więcej dziedzin (w przypadku baz danych rozważane są tylko relacje skończone (także zbiór pusty))

- iloczyn kartezjański: $D_1 \times D_2 \times ... \times D_k$ dziedzin $D_1, D_2, ... D_k$ to zbiór wszystkich k-krotek $(v_1, v_2, ... v_k)$ takich, ze v_1 należy do dziedziny D_1 , v_2 należy do dziedziny D_2 itd.

Krotki to elementy relacji (ozn. $(v_1, v_2, ... v_k)$, element v_i jest nazywany i-ta składową krotki)

Cechy relacji:

- nagłówek tabeli jest nazwą relacji (unikalną)
- każdy wiersz jest krotką
- krotki nie mogą się powtarzać
- porządek krotek nie jest istotny
- każda kolumna (atrybut) ma unikalną nazwę (ozn. A)
- porządek atrybutów nie jest istotny
- wszystkie składowe występujące w danej kolumnie mają wartości z tej samej dziedziny (ozn. Dom(A))
- liczba kolumn określa stopień relacji
- wszystkie krotki mają tą samą liczbę składowych występujących w tej samej kolejności

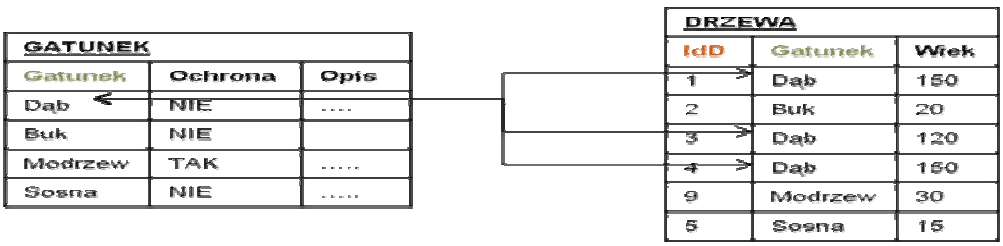
Klucz to atrybut (klucz prosty) lub zbiór atrybutów (klucz złożony) umożliwiający jednoznaczną identyfikację każdej krotki w relacji (żaden podzbiór klucza nie ma tej własności)

- rodzaje kluczy
  - klucz kandydujący – atrybut lub zbiór atrybutów jednoznacznie identyfikujący krotkę w relacji)
  - klucz główny (podstawowy) – klucz wyróżniony spośród kluczy kandydujących, musi być unikalny (unique) i nie może być pusty (NOT NULL)
  - klucz alternatywny – klucz kandydujący, który nie jest kluczem głównym
  - klucz obcy – klucz występujący w więcej niż jednej relacji i będący kluczem głównym w jednej z nich (musi nieć tą samą dziedzinę), jest wykorzystywany do ustalenia związku pomiędzy relacjami

Integralność danych

- integralność encji (związana z rozróżnialnością encji w zbiorze)
  - każda relacja posiadała klucz główny
  - każdy atrybut wchodzący w skład klucza głównego był unikalny i nie miał wartości NULL

- integralność referencyjna (związana z poprawnością powiązań pomiędzy encjami)
  - każda istniejąca, różna od NULL, wartość atrybutu składowego klucza obcego istniała także w relacji, w której jest ona składową klucza głównego



GATUNEK

Gatunek	Ochrona	Opis
Dąb	NIE	....
Buk	NIE	
Modrzew	TAK	.....
Sosna	NIE	.....

DRZEWA

IdD	Gatunek	Wiek
1	Dąb	150
2	Buk	20
3	Dąb	120
4	Dąb	150
9	Modrzew	30
5	Sosna	15

Więzy integralności gwarantują, że wpisane do bazy dane będą spójne.

Algebra relacji – definiuje zbiór operacji na danych i semantykę tych operacji

- operacje:
  - selekcja (SELECT) – wyznacza podzbiór krotek relacji, które spełniają określony warunek
  - rzutowanie, projekcja (PROJECT) – wyznacza podzbiór atrybutów relacji
    przykład: PROJECT EmpName (SELECT (EMP) DeptName = „Sales")

EMP

EmpNo	EmpName	DeptName	Grade
1	F Jones	Sales	6
2	P Smith	Accounts	6
3	K Chan	Sales	4
6	J Peters	Sales	5
9	S Abdul	Accounts	3
5	J Lewis	Reasearch	5

EMP'

EmpNo	EmpName	DeptName	Grade
1	F Jones	Sales	6
3	K Chan	Sales	4
6	J Peters	Sales	5

EMP"

EmpName
F Jones
K Chan
J Peters

  - suma (UNION) – wyznacza sumę teoriomnogościową relacji (tj. wszystkie krotki z dwóch kompatybilnych relacji z wykluczeniem duplikatów)
  - różnica (MINUS) – wyznacza różnicę teoriomnogościową relacji (tj. wszystkie krotki należące do pierwszej relacji i nie należące do drugiej)
  - przecięcie (INTERSECTION) – wyznacza część wspólną relacji
    przykład: FRENCH MINUS GERMAN

FRENCH

EmpNo	EmpName
2	P Smith
3	K Chen
6	J Peters
9	S Abul

GERMAN

EmpNo	EmpName
1	F Jones
3	K Chen
9	S Abul

FG

EmpNo	EmpName
2	P Smith
6	J Peters

  - złączenie (JOIN) – o umożliwia połączenie wybranych krotek z dwóch relacji w pojedynczą krotkę, na podstawie podanego warunku połączeniowego (nad atrybutami połączeniowymi, tj. atrybutami z różnych relacji mającymi tę samą dziedzinę)
    - złączenie naturalne (NATURAL JOIN) – na relacjach mających wspólny atrybut
    - złączenie zewnętrzne (OUTER JOIN)

    przykład: NATURAL JOIN (EMP, SKILL)

EMP

EmpNo	EmpName	DeptName	Grade
1	F Jones	Sales	6
2	P Smith	Accounts	6

SKILL

SEmpNo	Skill
1	Grman
2	Typing

EmpNo	EmpName	DeptName	Grade	SEmpNo	Skill
1	F Jones	Sales	6	1	Grman
2	P Smith	Accounts	6	2	Typing

przykład: NATURAL JOIN (EMP, SKILL) OUTER JOIN (EMP,COURSE) EmpNo = EmpNo

EMP

EmpNo	EmpName	Status
1	F Jones	10
2	P Smith	20
3	K Chen	10

EmpNo	EmpName	Status	CNo	CName	Skill
1	F Jones	10	1	French	1
2	P Smith	20	2	German	2
NULL	NULL	NULL	3	Spanish	NULL

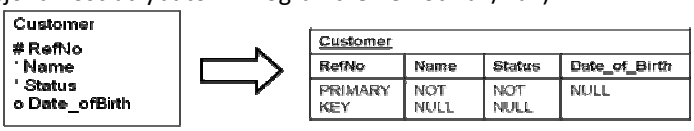
COURSE

CNo	CName	EmpNo
1	French	1
2	German	2
3	Spanish	NULL

PODSTAWY BAZ DANYCH – TRANSFORMACJA MODELU ER DO MODELU RELACYJNEGO

Krok1

- encja → relacja, klucz encji → klucz relacji, atrybuty encji → atrybuty relacji (obowiązkowość/opcjonalność atrybutów → ograniczenie not null/null)



Customer			
RefNo	Name	Status	Date_of_Birth
PRIMARY KEY	NOT NULL	NOT NULL	NULL

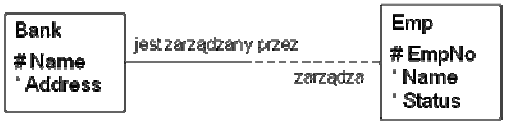
Krok2

- encja słaba → relacja, klucze główne encji od których zależy istnienie encji słabej → klucze obce

Realizacja					
funkcja	wynagrodzenie	od	do	idP	idPr
				PK, FK	PK, FK

Krok3 – związki

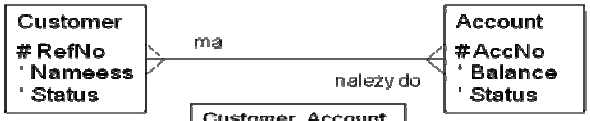
- 1:1: klucz główny relacji po stronie opcjonalnej → klucz obcy relacji po stronie obowiązkowej



Emp		
Name	Address	EmpNo
PK	NOT NULL	FK

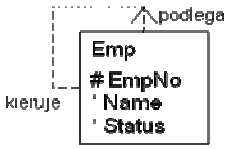
Emp		
EmpNo	Name	Status
PK	NOT NULL	NOT NULL

- 1:N: klucz główny relacji po stronie 1 → klucz obcy relacji po stronie N
- N:M: jak dla encji słabej



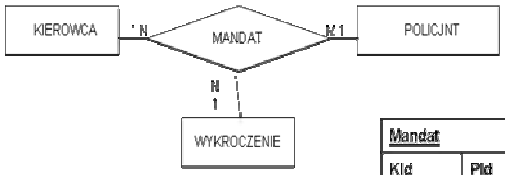
Customer_Account	
RefNo	AccNo
PK, FK	PK, FK

- unarny: klucz główny relacji → klucz obcy tej samej relacji



Emp			
NmpNo	Name	Status	MgrNo
PK			FK

- ternarny: jak dla encji słabej



Mandat		
Kid	Pid	Wid
PK, FK	PK, FK	PK, FK

Krok 4 – związki generalizacji

- nadencja i wszystkie podencja → jedna relacja zawierającą atrybuty wspólne i wszystkie atrybuty specyficzne
- każda potencja → odrębna relacja zawierająca wszystkie atrybuty wspólne i atrybuty specyficzne
- kadencja → relacja zawierającą atrybuty wspólne, każda podwncja → relacja zawierającą atrybuty specyficzne i klucz główny (nowy atrybut), klucze główne podencji → klucze obcymi nadencji

PODSTAWY BAZ DANYCH – NORMALIZACJA

Zależności funkcjonalne

- atrybut Y relacji jest funkcjonalnie zależny od atrybutu (podzbioru atrybutów) X tej relacji (ozn. X → Y), wtedy i tylko wtedy, gdy każdej wartości (kombinacji wartości) atrybutu X istnieje dokładnie jedna wartość (kombinacja wartości) atrybutu Y
  przykład: schemat relacji: Customer(RefNo, Name, Address, Status, AccNo)
  zależności: RefNo → Name, Address, Status; RefNo, AccNo → Name, Address, Status

- atrybut Y relacji R jest w pełni funkcjonalnie zależny od atrybutu złożonego X tej relacji jeżeli jest od niego zależny funkcjonalnie i nie jest zależny funkcjonalnie od żadnego podzbioru X
  przykład: w/w

- atrybut Y relacji R jest przechodnio funkcjonalnie zależny od atrybutu X tej relacji jeżeli X → Y i istnieje atrybut Z nie będący podzbiorem żadnego klucza, taki że zachodzi X → Z i Z → Y
  przykład: schemat relacji: Bank(BName, Address,Manager, AccNo, Balance, Type)
  zależności: AccNo → BName, Address, Manager, Balance, Type
  BName → Address, Manager – zależność przechodnia

Pierwsza postać normalna (1NF)

- relacja R jest w 1NF jeśli wartości wszystkich atrybutów R są atomowe (niepodzielne)

BANK					
BName	Address	Manager	AccNo	Balance	Type
Crawley	1, High Str.	1768	{120768, 678453, 348973}	{234.56, -456.78, 12567.56}	{'D', 'C', 'C'}
Bugstone	2, Low Str.	9823	{987654, 745363}	{789.65, -23.67}	{'C', 'D'}

relacja Bank nie jest w 1NF

BANK					
BName	Address	Manager	AccNo	Balance	Type
Crawley	1, High Str.	1768	120768	234.56	'D'
Crawley	1, High Str.	1768	678453	-456.78	'C'
Crawley	1, High Str.	1768	348973	12567.56	'C'
Bugstone	2, Low Str.	9823	987654	789.65	'C'
Bugstone	2, Low Str.	9823	745363	-23.67	'D'

relacja Bank jest w 1NF

Druga postać normalna (2NF)

- relacja R jest w 2NF jeśli jest w 1NF i wszystkie atrybuty niekluczowe (tj. nie będące częścią żadnego klucza kandydującego) są w pełni funkcjonalnie zależne od kluczy kandydujących relacji
- sprowadzenie relacji R do 2NF:
  - usunięcie z R atrybutów zależnych funkcjonalnie od podzbioru klucza kandydującego i utworzenia nowej relacji zawierającej usunięte atrybuty i podzbiór klucza, od którego są one zależne

CUSTOMER				
RefNo	Name	Address	Status	AccNo

relacja CUSTOMER nie jest w 2NF

RefNo, AccNo – klucz kandydujący
zależności: RefNo → Name, Address, Status; RefNo, AccNo → Name, Address, Status

CUSTOMER_ACC	
RefNo	AccNo

CUSTOMER			
RefNo	Name	Address	Status

Trzecia postać normalna (3NF)

- relacja R jest w 3NF jeśli jest w 2NF i nie zawiera zależności przechodnich
- sprowadzenie relacji R do 2NF:
  - usunięcie z R atrybutów zależnych przechodnio od atrybutu niekluczowego i utworzenia nowej relacji zawierającej usunięte atrybuty i atrybut, od którego są one zależne

BANK_ACC					
BName	Address	Manager	AccNo	Balance	Type

relacja BANK_ACC nie jest w 3NF

AccNo → Balance, Type, BName, Address, Manager; BName → Address, Manager (przechodnia)

BANK_ACC			
BName	AccNo	Balance	Type

BANK		
BName	Address	Manager

PODSTAWY BAZ DANYCH – SQL

SELECT

służy do pobierania i przetwarzania danych z relacji

- klauzula WHERE:
  - select * from customer where Status='business';
  - select Name, Address from customer where Status='business';
  - select * from customer where RefNo=2345;
- operatorLIKE:
  - select * from account where BranchName like 'C_a%';
    _ dowolny znak, % dowolny ciąg znaków
- operator BEWTEEN … AND:
  - select * from account where Balance between 0 and 1000 ;
- operator IN/NOT IN:
  - select * from account where Type in ('D', 'C')select * from customer where RefNo=2345;
- operator IS NULL/IS NOT NULL:
  - select * from customer where Address is not null;
- klauzula ORDER BY – sortowanie wyników:
  - select * from account order by Balance;
  - select * from account order by BranchName, Balance DESC;
    DESC – nierosnąco, ASC – niemalejąco (domyślne)
  - select * from account where Balance > 500 order by BranchName, ODate;
    klauzula order by musi być ostatnia
- aliasy kolumn i tabel:
  - select AccNo, (Balance+Balance*0.10) as 'new Balance' from account ;
  - select a.AccNo, a.Balance from account aselect * from account order by Balance;
- klauzula DISTINCT – usuwa duplikaty:
  - select distinct Typefrom account;

ZŁĄCZENIA

umożliwiają pobierania i przetwarzania danych z wielu relacji

- CROSS JOIN – zwraca iloczyn kartezjanski wybranych relacji
  - select * from customer cross join customer_accountselect Name, Address from customer where Status='business';
- INNER JOIN – zwraca krotki połączone względem atrybutu złączeniowego (dowolny warunek):
  - select * from customer c inner join customer_account ca on c.RefNo = ca.RefNo;
  - select * from customer c, customer_account ca where c.RefNo = ca.RefNo;
- NATURAL JOIN – zwraca krotki połączone względem domyślnego (taka sama nazwa w łączonych relacjach) atrybutu złączeniowego (równość wartości atrybutu):
  - select ca.AccNo, a.Balance from customer_account ca natural join account a where ca.RefNo = 8764 and a.Balance > 0;
  - select a.BranchName, a.AccNo, ca.RefNo from account a natural join customer_account ca where a.BranchName = 'Crawley;
- OUTER LEFT | RIGHT | FULL JOIN – zwraca krotki połączone względem atrybutu złączeniowego, umożliwia wyświetlenie także tych rekordów, dla których kryteria warunku złączeniowego nie są spełnione:
  - select c.RefNo, c.Name, ca.RefNo, ca.AccNo from customer c left join customer_account ca on c.RefNo = ca.RefNo;
  - select p.id_prac, p.Nazwisko, s.Nazwisko from pracownicy p left join pracownicy s on p.id_szefa = s.id_prac ;

PODZAPYTANIA

zapytania umieszczone wewnątrz innego zapytania (najczęściej w klauzuli where lub having)

- select distinct Name from customer where RefNo in (select RefNo from customer_account where AccNo in (select AccNo from account where BranchName = 'Crawley'))
  wierszowe – zwrcają pojedynczą wartość atrybutu lub grupy atrybutów
  tablicowe – zwracają podzbiór wartości atrybutu lub grupy atrybutów

- select Name from customer where (RefNo, Address) in (select RefNo,'23 High Str' from customer_account where AccNo in (select AccNo from account where BranchName = 'Crawley'));
- select * from customer_account where AccNo >= any (select AccNo from account where Balance > 0)
  operator ANY – warunek jest spełniony jeśli jest prawdziwy dla chociaż jednej wartości zwracanej przez podzapytanie
- select * from customer_account where AccNo >= all (select AccNo from account where Balance > 0)
  operator ALL – warunek jest spełniony jeśli jest prawdziwy dla wszystkich wartości zwracanych przez podzapytanie

PODSTAWY BAZ DANYCH – SQL

FUNKCJE

przekształcają pobierane dane i/lub pozwalają wyliczyć nowe wartości na podstawie pobranych danych

- podział funkcji
  - ze względu na zakres działania
    - wierszowe (testowe, numeryczne, daty i czasu, szyfrujące)
    - grupujące (agregujące)
  - ze względu na pochodzenie
    - wbudowane
    - podprogramy przechowywane
      - funkcje
      - procedury

FUNKCJE WIERSZOWE

tekstowe

NAZWA	OPIS
LENGHT(tekst)	Zwraca długość tekstu
LEFT(tekst, n)	Zwraca n pierwszyoh znaków
RIGHT(tekst,n)	Zwraca n ostatnich znaków
TRIM(tekst)	Usuwa początkowe i końcowe spacje
UPPER(tekst)	Zamienia tekst na duże litery
LOWER(tekst)	Zamienia tekst na małe litery
INITCAP(tekst)	Zamienia pierwszy znak na duży, pozostałe na małe
SUBSTRING(tekst, n, dl)	Zwraca dl znaków począwszy od n-tego
FIND_IN_SET(tekst, zbiór)	Zwraca 0 jeśli tekstu nie ma w zbiorze i >0 jeśli jest
CONCAT(tekst1, tekst2)	Łączy tekst1 z tekst2
REPLACE(tekst, t1,t2)	Zamienia w tekście ciąg t1 na ciąg t2
STRCMP(tekst1, tekst2)	Porównuje tekst1 z tekst2

- select c.Name, LEFT(c.Name,1) as First_Name, RIGHT(c.Name,LENGTH(c.Name)-2) as Name from customer c;
- select CONCAT(c.Name,' ', c.Address) as 'customer and address' from customer c

numeryczne

NAZWA	OPIS
ABS(n)	Wartość bezwzględna n
CEIL(n)	Zaokrąglenie do liczby całkowitej w górę
FLOOR(n)	Zaokrąglenie do liczby całkowitej w dół
MOD(n,m)	Reszta z dzielenia n przez m
POWER(n,m)	N do potęgi m
ROUND(n,m)	Zaokrągla n do m miejsc po przecinku
RAND()	Zwraca liczbę losową z przedziału 0 -1
SIGN(n)	Zwraca -1 jeśli n ujemne, 0 jeśli o i 1 jeśli dodatnie
TRUNCATE(n,m)	Obcina n do m miejsc po przecinku
FORMAT(n, m)	Formatuje do m miejsc po przecinku z krokami co 3 cyfry

- select a.Balance, round(a.Balance, 4), sign(a.Balance), ceil(a.Balance), floor(a.Balance) from account a;
- select a.Balance, format(a.Balance, 4), truncate(a.Balance,1) from account a;
- select a.Balance, concat(format(a.Balance, 2), ' ', 'PLN') as 'saldo w PLN' from account a

data i czas

NAZWA	OPIS
HOUR(data), MINUTE(data), ...	Zwraca liczbową wartość godziny, minuty, ...
MONTH(data), YEAR(data)	Zwraca liczbową wartość miesiąca, roku
DAYNAME(data), MONTHNAME(data)	Zwraca nazwę dnia, miesiąca
DATE_ADD(data, INTERVAL n)	Dodaje n jednostek do daty
DATE_SUB(data, INTERVAL n)	Odejmuje n jednostek od daty
CURDATE(), CURTIME()	Zwraca bieżącą datę, czas
NOW()	Zwraca bieżącą datę i czas
UNIX_TIMESTAMP(data)	Zwraca liczbę sekund od 1.1.1970 do daty
CONVERT_TZ(data, strefa1, strefa2)	Konwertuje datę ze strefy 1 do strefy 2

- select a.ODate, CURDATE(), YEAR(a.Odate), MONTH(a.Odate) from account a;

- select * from account a where DATE_ADD(a.ODate, INTERVAL 10 YEAR) < CURDATE();
- select UPPER(c.Name) as Name, a.AccNo, a.ODate, ca.RefNo from customer c, account a, customer_account ca where c.RefNo = ca.RefNo and a.AccNo = ca.AccNo and YEAR(a.ODate) >'2000';

szyfrujące

NAZWA	OPIS
MD5()	Zwraca skrót 32 znakowy
SHA1()	Zwraca skrót 40 znakowy
AES_ENCRYPT()	Szyfruje za pomocą algorytmu AES
AES_DECRYPT()	Deszyfruje za pomocą algorytmu AES
DES_ENCRYPT()	Szyfruje za pomocą algorytmu DES
DES_DECRYPT()	Deszyfruje za pomocą algorytmu AES

- select AES_ENCRYPT(Name, 'klucz') from custome;
- select AES_DECRYPT(AES_ENCRYPT(Name, 'klucz'), 'klucz');

inne

NAZWA	OPIS
CAST(wyr as typ)	Zmienia typ wyrażenia
CASE WHEN war1 THEN wyr1 WHEN war2 THEN wyr2 ELSE wyr3 END	
IF (war, wyr1wyr2)	

- select AccNo, case when Balance > 1000, '*********' , cast(Balance as char(8)) end from account ;
- select AccNo, if(Balance > 1000 , '*********', Balance) from account ;

FUNKCJE GRUPUJĄCE

NAZWA	OPIS
COUNT(*), COUNT(kol)	Zlicza wiersze
MIN(wyr), MAX(kol)	Znajduje najmniejszą/największą wartość w kolumnie
SUM(kol)	Sumuje wartości w kolumnie
AVG(kol)	Znajduje średnią wartości w kolumnie
GROUP_CONCAT(kol)	Złożenie wartości

zwykle są stosowane z klauzulą GRUP BY, może też występować z klauzula HAVING

- select count(*) from account; (*liczba kont założonych w banku)*
- select BranchName, count(*) from account group by BranchName;
- select BranchName, count(*) from account group by BranchName HAVING count(*) > 3;
- select * from customer where RefNo in (select RefNo from customer_account group by RefNo having count(*) > 1) ;
- select c.RefNo, c.Name, count(*) from customer c join customer_account ca on c.RefNo = ca.RefNo group by c.Name;
- select AccNo, Balance from account where Balance = (select max(Balance) from account) ;
- select RefNo, Name from customer natural join customer_account where AccNo = (select AccNo from Account where Balance = (select max(Balance) from account));
- select Name from customer where RefNo in (select RefNo from customer_account where AccNo = (select AccNo from account where Balance = (select min(Balance) from account))) ;
- select AccNo, Balance from account where Balance between (select 0.1*avg(Balance) from account) and (select 0.5*avg(Balance) from account) ;
- select BranchName, sum(Balance) as 'income' from account group by BranchName ;
- select group_concat(c.Name), a.BranchName from customer c, customer_account ca, account a where c.RefNo = ca.RefNo and ca.AccNo = a.AccNo group by a.BranchName ;

PODSTAWY BAZ DANYCH – SQL

DEFINIOWANIE DANYCH

CREATE DATABASE nazwa_bazy_danych; - tworzy bazę danych

- create database bank2;

DROP DATABASE nazwa_bazy_danych; - usuwa bazę danych

- drop database bank2;

CREATE TABLE (art1 typ [ograniczenia], atr2 typ [ograniczenia], …, [ograniczenia]); – tworzy schemat relacji

- create table customer (RefNo int(6) primary key, Name varchar(20) not null, Address varchar (30) null, Status char(8));
  ograniczenia atrybutów:
  - PRIMARY KEY – klucz główny relacji (jeśli prosty)
  - NOT NULL/ NULL – wartość nie musi/musi być podana
  - UNIQUE – wartość musi być unikalna w całej relacji
  - UNSIGNED – bez znaku
  - AUTO_INCREMENT – automatyczne ustalanie wartości (mysql)
  - DEFAULT *wartość* – wartość domyślna
  - CHECK *warunek* – ograniczenie dziedziny
- create table customer_account (RefNo int(6), AccNo int(6), primary key(RefNo, AccNo), index(RefNo), constraint fk_customer_RefNo foreign key(RefNo) references customer(RefNo), index(AccNo), constraint fk_account_AccNo foreign key(AccNo) references account(AccNo)) engine =innodb;
  ograniczenia dla relacji:
  - PRIMARY KEY(atr1, atr2,…) – stosowane jeśli klucz jest złożony
  - FOREIGN KEY(atr1,…) REFERENCES nazwa_relacji(atr1', …) – ustala związek pomiędzy wartościami atr1, .. bieżącej relacji i atr1', … relacji, w której atr1'. … są kluczem głównym (w mySQL relacje muszą być tylu InnoDB)
  - INDEX(atr1, …) – dodaje indeks do atrybutów

ALTER TABLE nazwa_relacji opis zmian – umożliwia dodawanie, usuwanie i modyfikowanie atrybutów i ograniczeń oraz modyfikowanie definicji atrybutów i relacji

- opis zmian dotyczących atrybutów
  - ADD COLUMN nazwa_atr typ_atr opis – umieszcza na końcu nową kolumnę
  - CHANGE COLUMN nazwa_art nowa_nazwa_art nowy_typ_atr opis
  - DROP COLUMN nazwa_atr – usuwa kolumnę
- opis zmian dotyczących ograniczeń
  - ADD CONSTRAINT nazwa_ogr definicja_ogr – nakłada ograniczenie na relację
  - ADD PRIMARY KEY (atr1,…) – tworzy klucz główny relacji
  - DROP PRIMARY KEY – usuwa klucz główny relacji
  - DROP FOREIGN KEY nazwa_ogr – usuwa klucz obcy
  - DROP KEY nazwa _ogr – usuwa ograniczenia (unique)
- alter table customer1 change column nowa nowa int not null;
- DROP COLUMN alter table customer drop column nowa;
- alter table customer_account add constraint pk_RefNo_AccNo primary key(RefNo, AccNo);
- alter table customer_account add primary key(RefNo, AccNo);
- alter table customer_account add index(RefNo);
- alter table customer_account add constraint fk_customer_RefNo foreign key (RefNo) references customer (RefNo);

CREATE VIEW nazwa_perspektywy (art1, atr2, …) AS zapytanie; – tworzy perspektywę (obiekt bazy danych będący nazwanym wynikiem zapytania, przechowywany jest tylko schemat perspektywy, dane są generowane gdy zadawane jest pytanie – zmiany w relacji bazowej perspektywy są także widoczne w perspektywie)

- create view c1(RefNo, Name, AccNo) as select c.RefNo, c.Name, ca.AccNo from customer c, customer_account ca where c.RefNo = ca.RefNo;

DROP TABLE nazwa_relacji; - usuwa relację
DROP VIEW nazwa_perspektywy; - usuwa perspektywę

MANIPULOWANIE DANYMI

INSERT INTO nazwa_relacji VALUES (v1, v2, ….) – wstawia krotkę do relacji
jeśli podawane są wartości dla wszystkich atrybutów (NULL jeśli nie podana)
wartości muszą być podane w zdefiniowanej kolejności

- insert into customer values(9419, 'D Johnson', '21 St Str', 'domestic');
- insert into customer values(9419, 'A Neil', NULL, 'domestic');

INSERT INTO nazwa_relacji (atr1, …) VALUES (v1, v2, ….) – wstawia krotkę do relacji
wymaga podania wartości tylko dla wymienionych atrybutów

- insert into customer (RefNo, Name, Status) values(5111, 'L Ullm', 'domestic');

UPDATE nazwa_relacji SET atr1 = v1, art2 = v2, …. – modyfikuje wartości wskazanych atrybutów

- update customer set Status = 'd' where Status = 'domestic';
- update customer set Status = 'b', RefNo = RefNo*10 where Status = 'business';
- update karta k set kara = 0.20*(select to_days('2010-04-23')-to_days(data_z) from karta_egz ke where k.id_czytelnik = ke.id_czytelnik);

DELETE FROM nazwa_relacji WHERE warunek – usuwa z relacji krotki spełniające warunek

- delete from customer where Addres is null;

TRUNCATE nazwa_relacji – usuwa wszystkie krotki z relacji

- truncate customer;

PODSTAWY BAZ DANYCH – SQL

PODPROGRAMY PRZECHOWYWANE

ZMIENNE

definicja: SELECT @zmienna := ‘wartość, SET @zmienna := ‘wartość’ lub SET @zmienna = ‘wartość’

- select @ts := avg(Balance) from account; select @l :=0.8;
  select AccNo, Balance from account where Balance between @ts-@l*@ts and @ts+@l*@ts;

INSTREUKCJE STERUJĄCE

- IF *warunek* THEN *instrukcje* ELSEIF *warunek1* THEN *instrukcje1 …* ELSEIF *instrukcje n* END IF
- CASE *warunek* WHEN *wartość1* THEN *instrukcje1 …* ELSEIF *instrukcje n* END CASE
- WHILE *warunek* DO *instrukcje* END WHILE
- REPEAT *instrukcje* UNTIL *warunek* END REPEAT
- LOOP *instrukcje* END LOOP

CREATE PROCEDURE nazwa_procedury(arg1 typ, arg2 typ, …) BEGIN instrukcje END – tworzy procedurę

- create procedure in_range(l float) begin
  select @ts := avg(Balance) from account; select AccNo, Balance from account where Balance between @ts-l*@ts and @ts+l*@ts; end//
- create procedure add_transaction(ANo int, amount float) begin
  if amount < 0 then set @type = 'out'; else set @type = 'in'; end if;
  insert into transaction (AccNo, TDate, Type, Amount) values(ANo,currdate(),@type, amount);
  update account set Balance = Balance + amount where AccNo = ANo; end//

CALL nazwa_procedury(arg1, arg2, …) – wywołuje procedurę

- call add_transaction(123456, 150);

WYZWALACZE

definiują akcje, które są wykonywane automatyczne, gdy w zdefiniowanej tabeli zajdzie zdefiniowane zdarzenie

CREATE TRIGGER nazwa_wyzwalacza czas zdarzenie ON nazwa_relacji FOR EACH ROW instrukcje

czas

BEFORE – jeśli wyzwalacz ma być wywołany przez wykonaniem zdarzenia
AFTER – jeśli wyzwalacz ma być wywołany przed wykonaniem zdarzenia

zdarzenie – odpowiada typowi akcji

INSERT – jeśli wiersz jest dodawany do tabeli
UPDATE – jeśli wiersz jest modyfikowany

- create trigger acc_trans after insert on transaction for each row begin
  set @amount = new.Amount; set @accno = new.AccNo;
  update account set Balance = Balance + @amount where AccNo = @accno;
  end;
- create trigger acc_trans2 before delete on transaction for each row begin
  if old.type = 'in' then set @amount = 0 - old.Amount; else set @amount = old.Amount; end if;
  set @accno = old.AccNo; update account set Balance = Balance + @amount where AccNo = @accno; end;

DROP TRIGGER nazwa_wyzwalacza – usuwa wyzwalacz

- create trigger acc_trans2;

PODSTAWY BAZ DANYCH – TRANSAKCJE

TRANSAKCJA

sekwencja logicznie powiązanych operacji na bazie danych, która przeprowadza bazę danych z jednego stanu spójnego w inny stan spójny

w MySQL

BEGIN – rozpoczęcie transakcji
//operacje wewnątrz transakcji
update konto set Balance = Balance – 100 where AccNo = A;
update konto set Balance = Balance + 100 where AccNo = B;
COMMIT – zakończenie transakcji i trwałe zapisanie zmian (zmiany są widoczne dla innych transakcji) lub
ROLLBACK – zakończenie transakcji i wycofanie wszystkich zmian

Zasady ACID – własności, jakie powinna mieć transakcja

atomowość (atomic)

zbiór operacji wchodzących w skład transakcji jest niepodzielny, tzn. zawsze wykonane zostaną wszystkie operacje transakcji albo żadna

spójność (consistent)

transakcja przeprowadza bazę danych z jednego stanu spójnego do innego stanu spójnego

izolacja (isolated)

transakcje są od siebie logicznie odseparowane mimo współbieżnego wykonywania

trwałość (durable)

wyniki zatwierdzonych transakcji nie mogą zostać utracone w wyniku wystąpienia awarii systemu

Poziomy izolacji

zjawiska niepożądane

niespójność odczytów – transakcja T2 może odczytywać zmieniane przez transakcje T1, chociaż T1 nie zatwierdziła zmian (tabela typu InnoDB nie pozwala na niespójność odczytów)
niepowtarzalność odczytów – zachodzi, gdy dwa kolejne odczytu tych samych danych przez transakcję T2 są różne
odczyty fantomowe – występują podczas dodawania danych, gdy transakcja T1aktualizuje dane i transakcja T2 widzi zmieniające się dane
utracone aktualizacje – zachodzi, gdy do bazy danych wprowadzane

definicje ANSI/ISO

anomalie / poziom izolacji	niespójność odczytów	niepowtarzalność odczytów	odczyty fantomowe
read uncommited	dopuszczalne	dopuszczalne	dopuszczalne
read commited	niedopuszczalne	dopuszczalne	dopuszczalne
repetable read	niedopuszczalne	niedopuszczalne	dopuszczalne
serializable	niedopuszczalne	niedopuszczalne	niedopuszczalne

Obsługa transakcji współbieżnych

Blokowanie

mechanizm wykorzystywany do zapobiegania anomalią, polega na przydzieleniu określonego zasobu do zadania
rodzaje blokad:
wyłączna – zasób jest przydzielony do zadania na wyłączność
współdzielona – jednocześnie kilka zadań może uzyskać dostęp do zasobu
zadanie = transakcja, zasób = relacja, wiersz, atrybut

BLOCK TABLES nazwa_relacji [WRITE | READ] – blokuje relację
READ – blokada do odczytu (nie zezwala na zmiany w tabeli), ale inni użytkownicy też mogą odczytywać dane
WRITE – blokada do zapisu (na wyłączność dla jednego użytkownika)

SELCECT …… [BLOCK IN SHARE MODE | FOR UPDATE] – blokuje wiersz wskazany warunkiem polecenia select

BLOCK IN SHARE MODE – blokada do odczytu

FOR UPDATE – blokada na wyłączność

Transakcja 1	Transakcja 2
lock tables account WRITE, customer_account READ, customer READ; update account set Balance = Balance + 100 where AccNo = A unlock tabels;	select RefNo, AccNo from customer_account where AccNo = A; +———+———+ | RefNo | AccNo | +———+———+ | 2345 | 1234567 | +———+———+ select Balance from account where AccNo = A; sesja zablokowana

Transakcja 1	Transakcja 2
select Balance from account where AccNo = B flock in shere mode; select Balance from account where AccNo = A for update; commit;	select Balance from account where AccNo = B; +———+ | Balance | +———+ | 200.00 | +———+ select Balance from account where AccNo = A; sesja zablokowana +———+ | Balance | +———+ | 400.00 | +———+

Zakleszczenie

ma miejsce gdy dwie transakcje czekają wzajemnie na zwolnienie zasobów i żadna transakcja nie może kontynuować swojego działania (mechanizm wykrywania zakleszczeń zakończy jedną z transakcji)

Transakcja 1	Transakcja 2

begin
update account set Balance = Balance - 100 where AccNo = A;
 	
update account set Balance = Balance + 100 where AccNo = B;
	begin
	update account set Balance = Balance - 200 where AccNo = B;
	
	update account set Balance = Balance + 200 where AccNo = A;

PODSTAWY BAZ DANYCH – UŻYTKOWNIKCY I UWIERZYTELNIANIE

UŻYTKOWNIK

osoba lub aplikacja, która jest uprawniona do dostępu do danych zgromadzonych w bazie danych

UWIERZYTELNIENIE

proces weryfikacji tożsamości użytkownika, który chce uzyskać dostęp do bazy danych

AUTORYZACJA

proces weryfikacji uprawnień użytkownika do realizacji określonych operacji w bazie danych

PRZYWILEJ/UPRAWNIENIE

prawo do wykonania określonej akcji w bazie danych lub możliwość dostępu do określonego obiektu w bazie danych

uprawnienia w MySQL

dotyczące danych

UPRAWNIENIE	OPIS
SELECT	odczytywanie wierszy relacji
INSERT	dodawanie nowych wierszy do relacji
UPDATE	aktualizowanie wierszy relacji
DELETE	usuwanie wierszy z relacji
FILE	dostęp do plików zapisanych na serwerze

dotyczące struktury

UPRAWNIENIE	OPIS
CREATE	tworzenie relacji
ALTER	modyfikowanie struktury relacji
INDEX	tworzenie i usuwanie indeksów
DROP	usuwanie tabel, baz danych
CREATE VIEW	tworzenie perspektyw
CREATE ROUTINE	tworzenie podprogramów przechowywanych
ALTER ROUTINE	modyfikowanie podprogramów przechowywanych
EXECUTE	wykonywanie procedury przechowywanej
TRIGGER	tworzenie i usuwanie wyzwalaczy

dotyczące administrowania

UPRAWNIENIE	OPIS
GRANT	tworzenie użytkownika i nadawanie uprawnień
SUPER	zakończenie pracy procesu
PROCESS	przeglądanie informacji o serwerze
RELOAD	powtórne załadowanie informacji z tabel uprawnień
SHUTDOWN	zatrzymanie bazy danych
CREATE USER	utworzenie użytkownika
LOCK TABLES	zablokowanie relacji
SHOW DATABASES	wyświetlanie dostępnych baz danych

GRANT prawa ON baza_danych.relacja TO nazwa_użytkownika [IDENTIFIED BY hasło] [WITH GRANT OPTION] – tworzy użytkownika i nadaje mu określone uprawnienia

- ON *.* - prawa do wszystkich baz danych i wszystkich tabel
- ALL – wszystkie prawa oprócz GRANT

- grant select, insert, update, delete on bank2.* to bankier@localhost identified by ‘money’
- grant select on firma.pracownicy to kowalski’%firma.eu’;

REVOKE prawa FROM nazwa_użytkownika – odbiera prawa użytkownikowi

- revoke insert, update, delete on bank2.* from praktykant@’%bank.pl’;

DROP nazwa_użytkownika – usuwa użytkonika

- drop user kowalski@localhost;

SET PASSWORD FOR nazwa_użytkonika = PASSWORD(‘hasło’);

PODSTAWY BAZ DANYCH – MYSQL I PHP

Podstawy składni

Znaczniki – <?php ….. ?> informują, że wszystko pomiędzy nimi jest kodem php

- **<?php** echo 'Hello world! jest już '; echo date('H:i:s'); **?>**

Zmienne

- <?php
  $liczba = 5; //integer
  $nazwa = "php";//string
  echo "$liczba"." "."$nazwa"; //konkatenacja ciągów
  ?>
  - typ zmiennej jest ustalany w oparciu o kontekst w jakim występuje (nie ustala się go jawnie!); funkcja gettype() zwraca typ zmiennej, funkcja settype() ustawia typ zmiennej

Przetwarzanie liczb i ciągów znaków

- <?php
  $liczba = 5; //5
  $liczba2 = $liczba + "-5.5"; //-0.5
  $liczba3 = $liczba + "10 liczb"; //15
  $liczba4 = $liczba + "liczba"; //5
  ?>
- jeśli ciąg znaków reprezentuje liczbę to wynikiem jest liczba
- jeśli ciąg znaków rozpoczyna się od liczby to tylko liczby są przekształcane, reszta ignorowana
- jeśli ciąg znaków rozpoczyna się od innych znaków to wynikiem jest 0

- <?php
  $dzien6 = "sobota";
  $dzien2 = 'wtorek';
  echo "$dzien6 $dzien2"; // sobota wtorek
  echo '$dzien6 $dzien2'; //$dzien6 $dzien2
  ?>
- apostrofy – całość jest traktowana jak tekst
- cudzysłów – rozpoznaje znaki specjalne (aby zdjąć znaczenie specjalne ze znaku należy go poprzedzić znakiem \)

Instrukcje sterujące – instrukcja warunkowa (if)

- <?php $haslo = "tajne!";
  if ($haslo == "tajne!") echo "Haslo poprawne";
  else echo "Haslo bledne, brak dostepu";
  ?>

Instrukcje sterujące – instrukcja wyboru (switch)

- <?php $nazwa = "Anna";
  switch($nazwa){
  case 2+3: echo " to jest liczba"."$nazwa"; break; //dopasuje 5 i „5”
  case "Anna":
  case "Tomasz": echo "to jest imie"; break;
  default: echo "nieznana nazwa"; }
  ?>

Instrukcje sterujące – instrukcja pętli (while)

- <?php $i = 1;
  while ($i<=10){
  $sq = $i*$i;
  echo " pierwiastek kwadratowy z $i wynosi $sq ";
  $i = $i +1; }
  ?>

Instrukcje sterujące – instrukcja pętli (do … while)

- <?php $i = 1;
  do {
  $sq = $i*$i;
  echo " pierwiastek kwadratowy z $i wynosi $sq ";
  $i = $i +1;
  } while ($i<=10);
  ?>

Instrukcje sterujące – instrukcja pętli (for)

- <?php
  for ($i=1; $i<=10;$i++){
  $sq = $i*$i;
  echo " pierwiastek kwadratowy z $i wynosi $sq ";}
  ?>
- we wszystkich rodzajach pętli można używać instrukcji break, continue

Funkcje

- <?php
  function after_tax($amount, $tax=22)
  {
  $total = $amount - $amount*$tax;
  return $total;
  }
  echo "jesli masz 1000 zl to zostanie ci ".after_tax(1000.00);
  ?>

  nie muszą być deklarowane przed użyciem, nie można przeciążać, można definiować argumenty domyślne, zmienne używane wewnątrz funkcji są w niej lokalne, dostęp do zmiennych globalnych wymaga jawnego przesłonięcia zasięgu poprzez umieszczenie wewnątrz funkcji konstrukcji: *global $nazwa_zmiennej;*

Tablice

indeksowane

- <?php
  $stolice = array("Berlin","Rzym","Warszawa","Ateny");
  $stolice[4] = "Madryt";
  echo $stolice[1]; //Rzym
  print "<PRE>"; print_r($stolice); print "<PRE>";
  ?>

asocjacyjne

- <?php
  $stolice = array("Wlochy"=>"Rzym","Polska"=>"Warszawa","Grecja"=>"Ateny");
  $stolice["Hiszpania"] = "Madryt";
  echo $stolice["Wlochy"];
  print "<PRE>"; print_r($stolice); print "<PRE>";
  ?>

  tablice są indeksowane od 0
  funkcja array() tworzy tablicę na podstawie podanej listy wartości (może być też użyta do zainicjowana tablicy: $stolice=array();)
  tablice mogą być indeksowane ciągami znaków (używa się ich tak samo jak tablic indeksowanych liczbami)

przetwarzanie tablic (pętle)

- <?php //$stolice ….
  while(list($key,$value) = each($stolice))
  **echo "stolica $key to $value
";**
  foreach($stolice as $key => $value)
  **echo "stolica $key to $value
";**
  ?>

Dostęp do bazy danych

połaczenie z baząp

```
<?php
$host_name="localhost";  $user_name="root";  $db_name="bank2";
$db=mysql_connect($host_name,$user_name);
mysql_select_db($db_name, $db);
// operacje na bazie danych
mysql_close($db);
?>
```

funkcja mysql_connect() – zwraca identyfikator łącza z bazą danych (false jeśli połączenie się nie uda)
jeśli istnieje hasło dla użytkownika to jest trzecim argumentem
funkcja mysql_select_db() – wybór bazy danych
funkcja mysql_close() – zamyka połączenie z bazą danych

wykonywanie zapytań

pobieranie danych

```
$sql_query = "select * from customer";
$sql_result=mysql_query($sql_query,$db);
for($i=0;$i<mysql_num_rows($sql_result);$i++)
{
   echo "klient $i: ";
   for($j=0;$j<mysql_num_fields($sql_result);$j++)
      echo mysql_result($sql_result, $i,$j)." ";
   echo "<br>";
}
```

funkcja mysql_query() – powoduje wysłanie zapytania i zwraca jego wynik
funkcja mysql_result() – pobiera element wyniku z określonego wiersza i kolumny
funkcja mysql_num_rows() – zwraca liczbę wierszy w wyniku
funkcja mysql_num_fields() – zwraca liczbę kolumn w wyniku (count() ma taki sam efekt)

```
$sql_query = "select RefNo, Name from customer";
$sql_result=mysql_query($sql_query,$db);
while($sql_row = mysql_fetch_array($sql_result))
{
  echo "RefNo: ".$sql_row["RefNo"]." "."Name: ".$sql_row["Name"]
  echo "<br>";
}
```

funkcja mysql_fetch_array() – zwraca wiersz z wyniku poleceń (false gdy nie ma więcej wierszy do pobrania), obsługuje obydwa rodzaje indeksowania (także mieszane)
podobnie działające funkcje:

mysql_fetch_row() – obsługuje tylko indeksy numeryczne
mysql_fetch_assoc() – obsługuje tylko indeksy asocjacyjne

```
$sql_query = "select RefNo, Name from customer";
$sql_result=mysql_query($sql_query,$db);
for($j=0;$j<mysql_num_fields($sql_result);$j++)
  echo mysql_field_name($sql_result,$j)."  ";
echo "<br>";
while($sql_row = mysql_fetch_array($sql_result))
{
  echo "RefNo: ".$sql_row["RefNo"]." "."Name: ".$sql_row["Name"]
  echo "<br>";
}
```

funkcja mysql_field_name() – zwraca nazwy pól

obsługa błędów

```
$sql_query = "select RefNo, Name from customer";
$sql_result=mysql_query($sql_query,$db);
if(!sql_result)
{
echo mysql_error()."<br>";
echo mysql_errno()."<br>";
exit;
}
```

funkcja mysql_error() – zwraca komunikat MySQL
funkcja mysql_errno() – zwraca numer błędu

proste wstawianie danych

```
$sql_query = "insert into customer values(2222,'A Adams','6 Str','b');";
$sql_result=mysql_query($sql_query,$db);
echo "dodano ".mysql_affected_rows($db)." wierszy";
$sql_query = "delete from customer where RefNo = 2222);";
$sql_result=mysql_query($sql_query,$db);
echo "usunieto ".mysql_affected_rows($db)." wierszy";
```

funkcja mysql_affected_rows() – zwraca liczbę wierszy przetworzonych w ostatnim poleceniu insert, delete i update

wstawianie danych z wykorzystaniem formularzy

<form action = „plik.php" method ="post"></form>

- określa obszar strony www, która zawiera elementy sterujące wprowadzaniem danych
- dane są przesyłane na adres URL (pełny adres, ścieżka względna do skryptu) określony przez atrybut ACTION
- sposób grupowania danych do przesłania przez przeglądarkę jest określony przez atrybut METHOD

prosty formularz do dodawania klientów

```
<form action = „dodaj.php" method ="post">
<table>
<tr><td>RefNo</td><td><input type = "text" name = "RefNo" size =30></td></tr>
<tr><td>Name</td><td><input type = "text" name = "Name" size =30></td></tr>
<tr><td>Address</td><td><input type = "text" name = "Address" size =30></td></tr>
<tr><td>Status</td>
<td><select name = "Status„>
<option value = "b">b</option>
<option value = "d">d</option></select>
</td></tr></table>
<input type = "submit" name = "dodaj" value = "dodaj">
</form>
```

inne rodzaje elementów

```
<input type = „radio" name = „wiek” value =”<10” >
<input type = „radio" name = „wiek” value =”1 - 18” >
<input type = „radio" name = „wiek” value =”>18” checked>
```

checkbox – pole wyboru do prowadzenia jednej z grupy możliwych wartości
wiek – nazwa pola, musi być taka sama dla całej zdefiniowanej grupy

```
<textarea rows=5 cols=20 name="komentarz">tu jest komentarz</textarea>
```

służy do tworzenia elementów tekstowych składających się z wielu wierszy

dostęp do danych wprowadzonych do formularza

```
echo "<PRE>";    print_r($_POST);    echo "<PRE>";
```

dane wprowadzone do formularza są dostępne poprzez tablicę superglobalną $_POST

```
$sql_query = "insert into customer values ('{$_POST["RefNo"]}',
'{$_POST["Name"]}','{$_POST["Address"]}','{$_POST["Status"]}');";
```