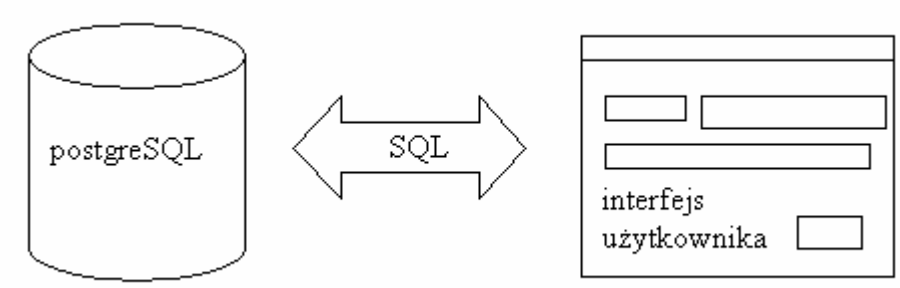
temat seminarium: „**Bazy danych – postgreSQL – programowanie i implementacja**”
temat pracy magisterskiej: „Baza danych postgreSQL – zarządzanie obrazami medycznymi”
27-listopad-2001
Jakub Turmiński

I Spis treści.

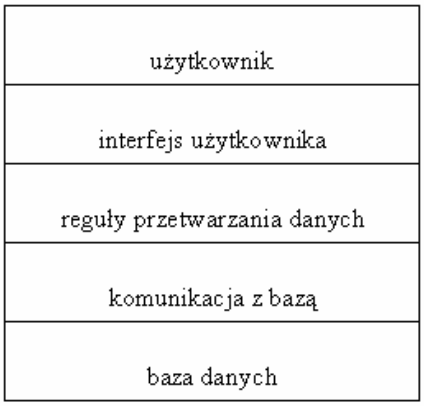
1. Wiadomości wstępne (pojęcia)
    - baza danych
    - DBMS
    - transakcja
    - RDBMS (relacyjne bazy danych) 1969r
    - Język zapytań (SQL) (standardy SQL-1 –1986, SQL-2 – 1992, SQL3-?)
    - Rodzaje baz danych (lokalne, klient-serwer)

2.1 Architektura aplikacji współpracującej z baza danych

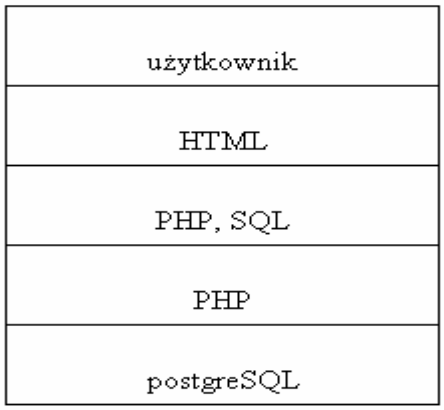


rys.1

2.2 Rysunek teoretyczny – warstwowy model architektury

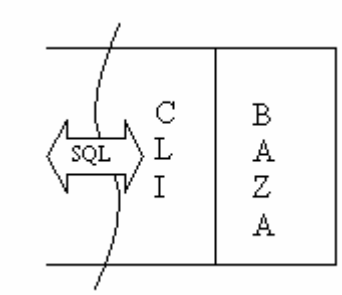


2.3 Rysunek warstwowy model architektury w naszym przykładzie

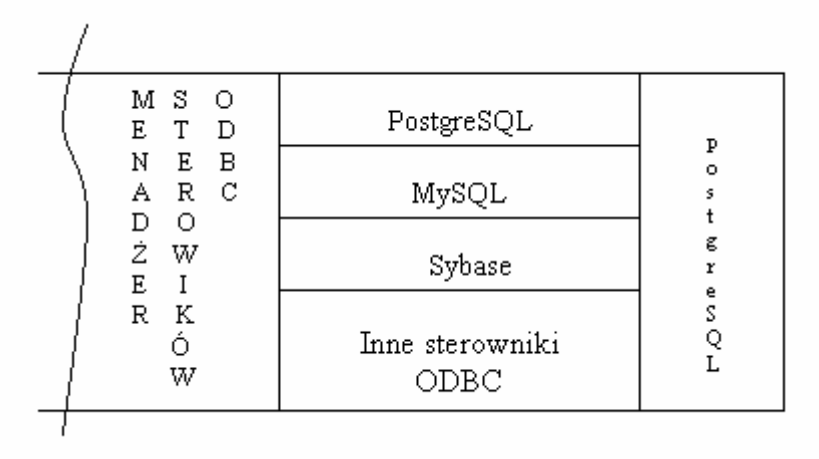


Rys.3

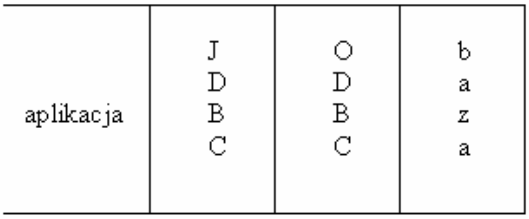
2.4 Aplikacja bazy danych jako interfejs użytkownika
2.5 Warstwa środkowa – reguły dziedziczenia danych
2.6 Komunikacja z bazą: sterowniki, dostęp do danych, protokoły sieciowe



2.7 Sterowniki ODBC (Microsoft)



2.8 Sterowniki JDBC (Sun)



3. Relacyjne bazy danych na przykładzie postgreSQL-a
3.1 PostgreSQL – jako relacyjna baza danych
3.2 PostgreSQL – jako baza danych w architekturze klient-serwer
3.3 Instalacja PostgreSQL-a
3.4 Demon bazy –> postmaster (uruchamianie: #postmaster –i & (port5432))
3.5 Zmienne środowiskowe ($PGHOST, $PGDATA, $PGUSER, $LOGFILE, $PGLIB)
3.6 Administracja postgreSQL-em
3.6.1 baza wzorcowa –template1
3.6.2 programy narzędziowe (createdb, destroydb, createuser, destroyuser, vacuumdb)
3.6.3 plik konfiguracyjny dostępu do baz danych postgreSQL-a – pga_hba.cfg
3.7 Interaktywne programy klienckie (psql, kpsql, pgaccess)

4 Język zapytań - SQL (przykłady)
4.1 CREATE DATABASE turmomed;
4.2 DROP DATABASE turmomed;
4.3 CREATE TABLE patient (

```
        id_patient SERIAL,
        surname VARCHAR(30),
        name VARCHAR(15),
        address VARCHAR(50),
        PRIMARY KEY(id_patient)
);
CREATE TABLE examination (
        id INTEGER NOT NULL UNIQUE PRIMARY KEY,
        nr_pics INTEGER,
        type varchar(5),
        id_patient INTEGER REFERENCE patient(id_patient)
);
```

4.4 INSERT INTO patient (surname, name, address)

VALUE (‘Kowalski’, ’Jan’, ’ul. Ciepła 5/3’);

4.5 UPDATE patient SET imie=’Piotr’
WHERE id_patient=9;

4.6 DELETE FROM patient
WHERE nazwisko = ‘Kowlaski’;

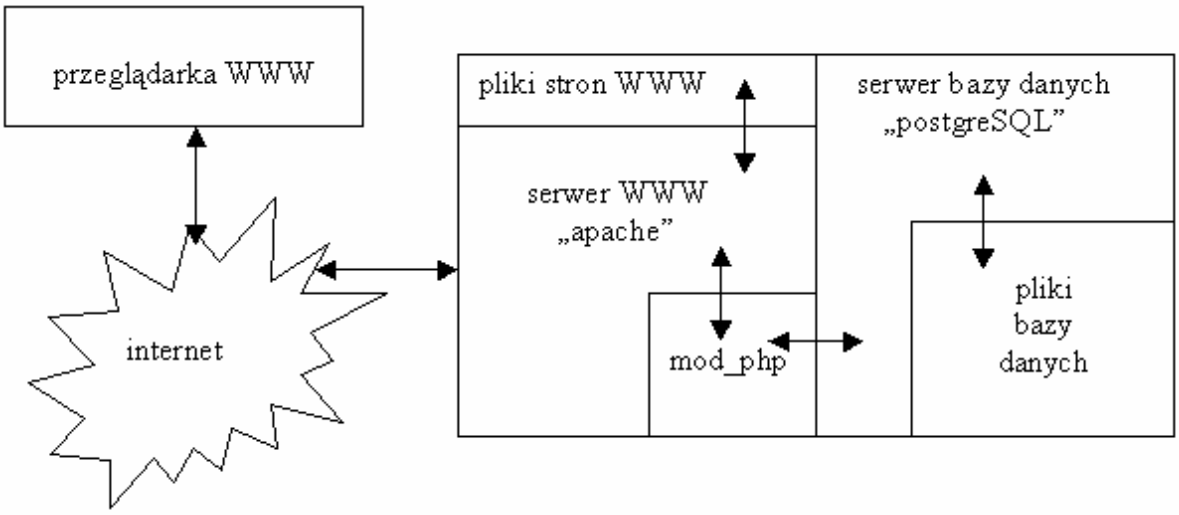
4.7 SELECT nazwisko, imie, adres
FROM patient
WHERE id_exam = 132;
SELECT surname, name, type
FROM patient JOIN examination
WHER id = 17;

4.8 CREATE INDEX surname_name
ON patient (surname, name);

5. Interfejsy bazy danych
- HEITML (rozszerzenie HTML’a o <IF>, <WHILE>, itd...)
- APPGEN 4GL
- Interface CGI/perl
- Python
- ODBC, JDBC
- PHP/FI

6. PHP/FI (funkcje dostępne do obsługi postgreSQL-a)

6.1 Schematyczny rysunek ułatwiający zrozumienie działania skryptów PHP



Rys.7 Sposób w jaki PHP współpracuje z przeglądarką, serwerm WWW oraz serwerem baz danych.

6.2 Znaczniki PHP (<? ?> <?PHP ?>)

6.3 Zmienne w PHP ($i=1;)

6.4 PHP jako uproszczony jesyk C++

6.5 Najważniejsze funkcje PHP służące do obsługi baz postgreSQL
- pg_connect() – funkcja otwiera połączenie do wskazanej bazy danych postgreSQL I w przypadku poprawnego wykonania zwraca identyfikator połączenia. W razie pojawienia się jakichkolwiek błędów funkcja zwraca wartość FALSE. Identyfikator zwrócony przez tę funkcję jest wykorzystywany w

wywołaniu innych funkcji służących do obsługi baz danych postgrSQL. Definicja funkcji wygląda jak następuje:

int pg_connect(string host, string port, string opcje,
string tty, string nazwa_bazy);

- pg_exec() – funkcja wykonuje zapytanie SQL na bazie danych postgreSQL określonej za pomocą identyfikatora połączenia (argument połączenie). Jeśli podane polecenie mogło być wykonane, funkcja zwróci identyfikator wyników. Jeśli identyfikator połączenia nie jest poprawny lub jeśli podczas wykonania funkcji pojawią się jakieś błędy, zwracana jest wartość FALSE. Funkcja zwraca indeks, który może zostać wykorzystany do pobrania wyników zapytania.
- pg_fieldisnull() – funkcja zwraca wartość TRUE, jeśli wartość pola wynosi NULL. W przypadku gdy wartość pola nie wynosi NULL, funkcja zwraca wartość FALSE. Pole można określić podając jego nazwę lub indeks. Numeracja wierszy rozpoczyna się od wartości 0. Definicja przedstawia się następująco:
  int pg_fieldisnull(int id_wyników, int wiersz, mixed pole);
- Pg_fieldname() – funkcja zwraca nazwę określonego pola ze zbioru wyników. Pole to określone jest za pomocą liczby, przy czym numeracja zaczyna się od 0. A oto składnia funkcji:
  String pg_fieldname(int id_wyników, int numer_pola);
- pg_fieldsize() – funkcja zwraca wielkość pola określonego za pomocą liczby. Jeśli funkcja zwróci wartość 0, będzie to oznaczało, iż długość pola może się zmieniać. W przypadku pojawienia się błędów zwracana jest wartość –1. Składnia jest następująca:
  int og_fieldsize(int id_wyników, int numer_pola);
- pg_fieldtype() – funkcja zwraca łańcuch znaków zawierający nazwę typu danych wskazanego pola zbioru wyników. Numeracja pól zaczyna się od 0. Składnia funkcji wygląda następująco:
  string pg_fieldtype(int id_wyników, int numer_pola);
- pg_numrows() – funkcja zwraca ilość wierszy w zbiorze wyników. Argument id_wyników musi być poprawnym identyfikatorem wyników zwróconym przez funkcję pg_exec(). W przypadku powstania jakichkolwiek błędów funkcja zwraca wartość –1. A oto składnia funkcji:
  int pg_numrows(int id_wyników);
- pg_numfields()– funkcja zwraca ilość pól w zbiorze wyników. Argument id_wyników musi być poprawnym identyfikatorem wyników zwróconym przez funkcję pg_exec(). W przypadku powstania jakichkolwiek błędów funkcja zwraca wartość –1. A oto składnia funkcji:
  int pg_numfields(int id_wyników);
- pg_result() – funkcja zwraca pojedynczą daną ze zbioru wyników o podanym identyfikatorze. Identyfikator zbioru wyników musi być zwrócony przez funkcję pg_exec(). Identyfikator wyników wskazuje na dane zapisane w wielu wierszach, z których każdy zawiera wiele pól. Argument numer_wiersza wskazuje, z którego wiersza ma pochodzić pobrane pole. Samo pole określane jest za pomocą argumentu nazwa_pola, może to być nazwa pola lub jego indeks. Zarówno indeksy wierszy jak i pól rozpoczynają się od wartości 0. Funkcja zwraca dane w postaci liczb całkowitych, zmiennoprzecinkowych lub łańuchów znaków. Definicja funkcji wygląda następująco:
  mixed pg_result(int id_wyników, int numer_wiersza, mixed nazwa_pola);

- pg_cmdtuples – funkcja zwraca ilość wierszy objętych działaniem polecenia SQL UPDATE, INSERT, bądź DELETE. Jeśli nie ma takich wierszy, funkcja zwróci wartość 0. Składni funkcji wygląda następująco:
  int pg_cmdtuples(int id_wyników);
- pg_close() – funkcja zamyka podane połączenie. W przypadku gdy argument połączenia nie jest poprawnym identyfikatorem połączenia, funkcja zwraca wartość FALSE. A oto składnia funkcji:
  bool pg_close(int połączenie);

7. Formularze HTML

Formularz HTML jest zestawem pól umożliwiających użytkownikowi wprowadzenie danych. Gdy użytkownik naciśnie przycisk „SUBMIT", wprowadzone dane zostaną wysłane określonemu w formularzu skryptowi PHP.

Standardowy HTML dostarcza do konstrukcji formularzy bogatego zestawu obiektów: pól tekstowych, przełączników, list i pól wyboru.

Treść formularza, czyli definicje pól do wprowadzania danych, zawarta jest pomiędzy dyrektywami <FORM> i </FORM>. Dyrektywa <FORM> ma dwa argumenty:

- METHOD. Definiuje metodę używaną do przekazywania danych skryptowi PHP. Metodą tą może być GET lub POST. Pierwsza z nich oznacza, że dane z formularza zostaną umieszczone na końcu adresu URL, które przeglądarka przesyła na serwer jako żądanie. Serwer WWW przekazuje następnie te dane do interpretera PHP lub programu CGI pod postacią zmiennej środowiskowej o nazwie QUERY_STRING. Druga wymusza użycie standardowego wejścia. W tym przypadku serwer WWW przesyła dane z formularza do skryptu PHP lub programu CGI w standardowym strumieniu wejściowym –STDIN.
- ACTION. Określa adres sieciowy skryptu PHP lub programu CGI, który ma zająć się obsługą przekazywanych danych.

Oto przykład użycia:

```
<FORM METHOD=POST ACTION="http://goblin.umcs.lublin.pl/~turmoll.index.phtml">
...
</FORM>
```

Między dyrektywami <FORM> i </FORM> znajdują się definicje elementów pozwalających na wprowadzanie danych. Każdy z nich określa osobne pole danych. Formularz może składać się z trzech rodzajów elementów, opatrzonych dyrektywami <INPUT>, <SELECT> i <TEXTAREA>.

a) Dyrektywa <INPUT> może przybierać osiem różnych postaci, z których każda reprezentuje jeden typ obiektu. Poniżej przedstawiamy poszczególne typy obiektów.
   - Pole wyboru (ang. checkbox) wyświetlane jest zwykle jako przycisk, który użytkownik może wcisnąć (oznaczając wybór pola) lub zwolnić (usuwając wybór). Oto składnia polecenia tworzącego pole wyboru:
     <INPUT TYPE=checkbox NAME=nazwa
     VALUE=wartosc [CHECKED]>
     Jeśli przycisk pola jest wciśnięty, nazwa pola ustawiona jest jako równa jego odpowiedniej wartości. Jeśli przycisk nie jest wciśnięty, przeglądarka nie wysyła opisujących to pole danych. Jeśli w deklaracji pola pojawi się opcjonalne słowo CHECKED, pole jest domyślnie zaznaczone.
   - Pole ukryte. (ang. hidden field) nie jest widoczne na ekranie. Tworzona jest następującym poleceniem:

<INPUT TYPE=hidden NAME=nazwa VALUE=wartosc>
Nazwa pola i jego wartość przekazywane są skryptowi PHP przy wysyłaniu formularza. Pola ukryte są najprostszym sposobem przechowania danych wprowadzonych we wcześniejszej sesji przez użytkownika pracującego z wieloma formularzami i jedyna praktycznie metodą na przekazanie danych z jednego skryptu PHP do innego (pamiętając oczywiście o cookies).

- Pole graficzne umożliwia wyświetlenie obrazka (ang. image), który użytkownik może traktować jako przycisk. Oto jak je utworzyć:
  <INPUT TYPE=image NAME=nazwa
  SRC=adres [ALIGN=[top | middle | bottom]]>
  Adres określa adres pliku graficznego do wyświetlenia.
  Gdy użytkownik kliknie myszką obrazek, przeglądarka prześle zawartość formularza skryptowi PHP określonemu dyrektywą FORM.
- Pole hasła są polami umożliwiającymi wprowadzenie hasła (ang. password) różnią się od opisanych niżej zwykłych pól tekstowych sposobem wyświetlania wpisanego tekstu – niezależnie od rodzaju wprowadzonego znaku wyświetlany zostanie zawsze ten sam znak (najczęściej gwiazdka). Pole to jest używane do wprowadzenia haseł i innych informacji, które muszą pozostać tajne. Oto składnia definicji pola:
  <INPUT TYPE=password NAME=nazwa
  [MAXLENGHT=długość SIZE=rozmiar
  VALUE=wartość_domyślna]>
  Opcjonalny argument MAXLENGTH umożliwia ustawienie maksymalnej długości łańcucha, jaki użytkownik może wpisać w tym polu. Argument SIZE określa rozmiar okna wyświetlanego na ekranie użytkownika. Jeśli rozmiar jest mniejszy niż długość łańcucha, pole zapewni przewijanie tekstu. Opcjonalny argument VALUE określa domyślną wartość pola hasła.
- Pole przełączników (ang. radio button) umożliwia zdefiniowanie grupy przycisków, z których tylko jeden może być wciśnięty – wybór jednego oznacza rezygnację z wyboru innych. A oto jak definiujemy grupę przełączników:
  <INPUT TYPE=radio NAME=nazwa VALUE=wartosc [CHECKED]>
  Każdy przycisk grupy posiada tę samą nazwę, każdy jednak ma osobną wartość. Przy wysyłaniu formularza wartość całego pola ustawiona jest jako wartość przycisku, który był wybrany w chwili wydania polecenia przesłania formularza.
- Przycisk RESET służy do skasowania wszystkich danych wprowadzonych przez użytkownika w formularzu. Oto definicja tego przycisku:
  <IMPUT TYPE=reset NAME=nazwa [VALUE=tekst]>
  Opcjonalny parametr VALUE umożliwia podanie wyświetlanej na przycisku etykiety.
- Przycisk SUBMIT jest używany do wysłania zawartości formularza skryptowi PHP. Oto jak definiujemy ten przycisk:
  <IMPUT TYPE=submit NAME=nazwa [VALUE=wartosc]>
  Opcjonalny parametr VALUE umożliwia podanie wyświetlanej na przycisku etykiety.
- Pole tekstowe umożliwia użytkownikowi wpisanie jednej linii tekstu. W przeciwieństwie do pola hasła , pole tekstowe wyświetla wprowadzony tekst w oryginalnej postaci. Oto składnia polecenia definiującego pole tekstowe.

<INPUT TYPE=text NAME=nazwa
[MAXLENGHT=długość SIZE=rozmiar
VALUE=wartość_domyślna]>

Opcjonalny argument MAXLENGHT umożliwia określenie maksymalnej długości łańcucha pobieranego przez pole. Argument SIZE określa rozmiar okna wyświetlanego na ekranie użytkownika. Jeśli rozmiar jest mniejszy niż długość łańcucha, pole zapewni przewijanie tekstu. Opcjonalny argument VALUE określa domyślną wartość pola tekstowego.

b. Dyrektywa <SELECT> umożliwia definicję listy wyboru, z której poprzez kliknięcie użytkownik może wybrać jedną z kilku wartości. Oto składnia dyrektywy tworzącej listę:

<SELECT NAME= nazwa [MULTIPLE SIZE=rozmiar]>
<OPTION [SELECTED VALUE=wartosc]> tekst_opcji </OPTION>
…
</SELECT>

Argument NAME przypisuje nazwę elementowi SELECT. Opcjonalny argument MULTIPLE dopuszcza wybór przez użytkownika więcej niż jednej wartości z listy. Argument SIZE określa liczbę wierszy listy wyświetlanych w oknie – jeśli liczba zdefiniowanych opcji przekracza liczbę widocznych wierszy, przeglądarka wyświetla zwykle pasek przewijania, umożliwiając przesuwanie zawartości listy. Pomiędzy dyrektywami <SELECT> i </SELECT> zawarte są dyrektywy <OPTION>, definiujące wartości wyświetlane na liście. Opcjonalny argument SELECTED wskazuje domyślny wybór opatrzonej nią wartością. Argument VALUE określa wartość przesłaną skryptowi PHP w przypadku wybrania przez użytkownika danej opcji – jeśli nie jest on ustawiony, przeglądarka przekazuje programowi tekst_opcji, zawarty w sekcji <OPTION>. Jeśli długość tekstu najdłuższej z podanych wartości przekracza rozmiar okna, które przeglądarka może przydzielić liście (może to zależeć od rozmiaru okna przeglądarki lub używanej czcionki), w zależności od przeglądarki może zostać wyświetlony poziomy pasek przewijania albo zawinięty lub obcięty tekst opcji.

c. Element TEXTAREA definiuje wielowierszowe pole tekstowe, do którego użytkownik może wpisać przesłane dane. Oto składnia definicji:

<TEXTAREA NAME = nazwa COLS = liczba_kolumn
ROWS=liczba_wierszy>
pierwsz linia tekstu
...
</TEXTAREA>

Tekst zawarty między dyrektywami <TEXTAREA> i </TEXTAREA> traktowany jest i wyświetlany jako domyślna wartość pola.