UTILIZAREA TEHNOLOGIEI CLIENT-SERVER ÎN INTEROGAREA BAZELOR DE CUNOȘTINȚE ÎN LIMBAJ NATURAL

Universitatea “SPIRU HARET”
Facultatea de Management Financiar-Contabil Craiova
Asist.univ.drd. Sorin Dincă

Interogarea în limbaj natural a bazelor de cunoştinţe

- Rețelele de tranziție pot fi utilizate cu succes în analiza structurii sintactice a unui text.
- O diagramă de tranziție este un graf orientat în care arcele sunt etichetate cu cuvinte ale unui limbaj natural. Nodurile grafului se numesc stări. Un arc orientat desemnează trecerea dintr-o stare în alta. O stare în care nu intră niciun arc se numeşte *stare inițială*. Un arc din care nu iese niciun arc se numeşte *stare finală*.
- O frază a unui limbaj natural este descompusă în *tokeni*, care separă un concept abstract de obiectele care sunt cazuri particulare ale acestui concept. Pentru a fi acceptată fraza, succesiunea de tokenuri trebuie să corespundă unui drum din diagramă, care începe cu o stare inițială şi se termină cu o stare finală.

- Puterea generativă a diagramelor de tranziție poate fi extinsă prin înlocuirea entităților arcelor cu categoriile sintactice care corespund acestora.
- Astfel se va obține o *rețea de tranziție de bază* - BTN (Basic Transition Network).
- Pentru a obține un BTN, se pot introduce următoarele comenzi pe arce:

  - *JUMP sk* – salt necondiționat de la o stare la alta;
  - *WRD w* – precizează faptul că tranziția respectivă trebuie să consume cuvântul w;
  - *CAT xzy* – specifică faptul că trebuie să se consume o entitate aparținând categoriei sintactice xzy.

- Rețelele de tranziție fiind echivalente cu automatele finite deterministe, nu pot acoperi toată clasa limbajelor independente de context. Pentru a realiza aceasta, se extinde definiția la rețele recursive de tranziție.
- O *rețea de tranziție recursivă* RTN (Recursive Transition Network) se obține dintr-o rețea de tranziție de bază BTN dacă dăm posibilitatea ca cel puțin un arc al acesteia să fie etichetat cu comanda: *PUSH nod*.
- Aşadar o rețea de tranziție recursivă se poate caracteriza prin ecuația:

RTN = BTN + PUSH

- Comanda PUSH nod realizează următoarele operații:
  - întrerupe operațiile aflate în curs de desfăşurare;
  - memorează starea actuală a rețelei;
  - trece la nodul **nod**; acest nod este considerat drept nod inițial pentru o altă rețea.

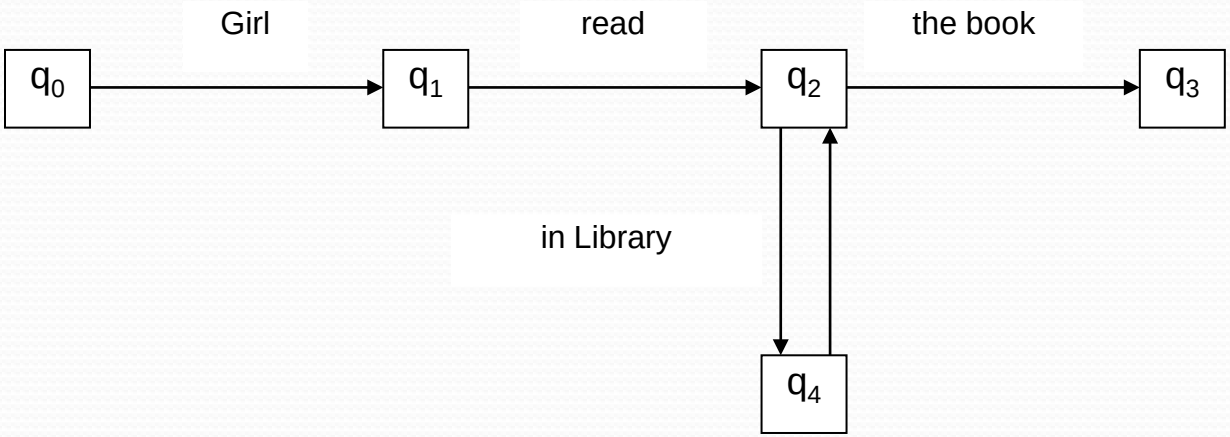

Fig. 1. Diagramă de tranziție

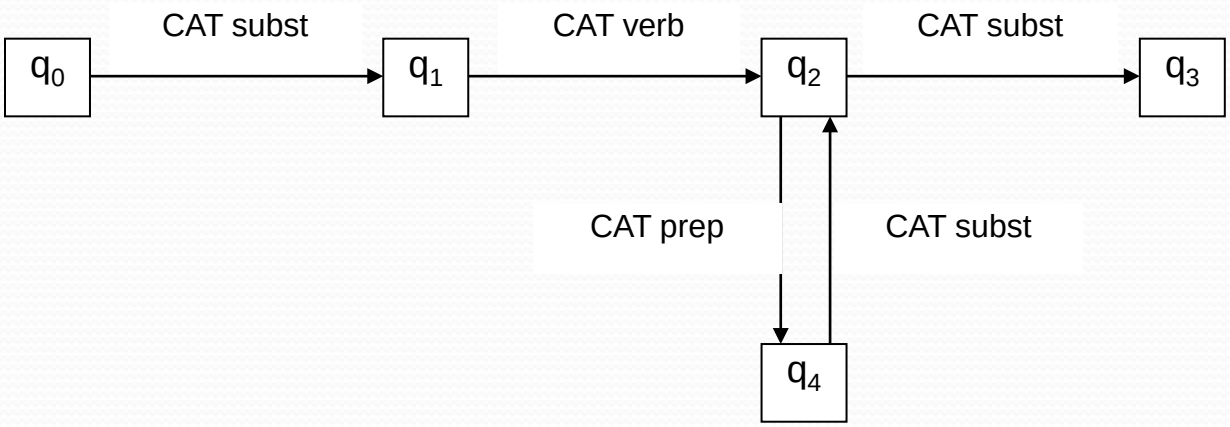

Fig. 2. Basic Transition Network (BTN)

Tehnologia client-server în interogarea bazelor de cunoştinţe

- Un protocol reprezintă o convenție de reprezentare a datelor folosită în comunicarea între două calculatoare.
- Cele mai utilizate protocoale sunt TCP (Transmission Control Protocol), şi UDP (User Datagram Protocol).
- Orice calculator conectat la Internet este identificat în mod unic de adresa IP (Internet Protocol).
- Un socket este un termen software abstract folosit pentru a reprezenta fiecare din cele două capete ale unei conexiuni între două procese ce rulează într-o rețea.
- O aplicație de rețea ce foloseşte socket-uri se încadrează în modelul client/server de concepere a unei aplicații.

- Programele de tip server sunt cele care oferă diverse servicii eventualilor clienți, fiind în stare de aşteptare atâta vreme cât nici un client nu le solicită serviciile. Programele de tip client sunt cele care inițiază conversația cu un server, solicitând un anume serviciu. Un server trebuie să fie capabil să trateze mai mulți clienți simultan, deci, fiecare cerere adresată serverului va fi tratată într-un fir de execuție separat.
- Avantajul protocolului TCP/IP este că asigură realizarea unei comunicări stabile, permanente în rețea, existând siguranța că informațiile trimise de un proces vor fi recepționate corect şi complet la destinație sau va fi semnalată o excepție în caz contrar.

Utilizare limbajului Java în tehnologia client-server

- Java este un limbaj de programare orientat obiect.
- Există mai multe tipuri de aplicații Java:
  a. aplicații de sine-stătătoare (stand-alone);
  b. aplicații care se execută pe partea de client (appleturi);
  c. aplicații care se execută pe partea de server (servleturi).
- Programarea cu fire de execuție este un aspect important al limbajului Java.
- Firele de execuție sunt cele care execută instrucțiunile. Firele de execuție nu pot rula decât în cadrul unui proces. Fiecare fir de execuție (*thread*) are o prioritate de execuție.

- **Structura generală a unui server bazat pe conexiuni este:**

  1. Creează un obiect de tip ServerSocket la un anumit port

  while(true) {

  2. Aşteaptă realizarea unei conexiuni cu un client, folosind metoda accept;

  (va fi creat un obiect nou de tip Socket)

  3. Tratează cererea venită de la client:

  3.1. Deschide un flux de intrare şi primeşte cererea

  3.2. Deschide un flux de ieşire şi trimite răspunsul

  3.3. Închide fluxurile şi socketul nou creat

  }

- **Structura generală a unui client bazat pe conexiuni este:**

1. *Citeşte sau declară adresa IP a serverului şi portul la care acesta rulează;*
2. *Creează un obiect de tip Socket cu adresa şi portul specificate;*
3. *Comunică cu serverul:*
    - *3.1. Deschide un flux de ieşire şi trimite cererea;*
    - *3.2. Deschide un flux de intrare şi primeşte răspunsul;*
    - *3.3. Închide fluxurile şi socketul creat.*

- Pentru fiecare din cele două socket-uri deschise pot fi create apoi două fluxuri pe octeți pentru citirea, respectiv scrierea datelor.

- Structura **JTable** este utilizată pentru a afişa şi edita tablouri cu două dimensiuni. Fiecare tabel utilizează un *table model object* pentru managementul datelor actuale ale unui tabel. Un table model object trebuie să implementeze interfața TableModel. Dacă programatorul nu specifică un anumit TableModel, JTable în mod automat creează o instanță a obiectului *DefaultTableModel*.
- Un obiect *Container* este o componentă care poate să conțină alte componente AWT (Abstract Window Toolkit).
- Metoda *setLayout* a obiectului Container este utilizată pentru a seta un manager de componente pentru acest container.
- Clasa *GridBagLayout* este un manager care aliniază componentele pe orizontală şi pe verticală, fără să ceară ca toate componentele să aibă aceeaşi dimensiune.
- Un obiect GridBagLayout aşează componentele după o matrice dreptunghiulară în care o componentă poate ocupa una sau mai multe celule ale matricii.

- Fiecare componentă manageriată de GridBagLayout este asociată cu o instanță a obiectului GridBagConstraints.
- Pentru a utiliza un GridBagLayout vor trebui particularizate unul sau mai multe obiecte GridBagConstraints asociate componentelor folosite.
- Particularizarea se face prin setarea variabilelor acestuia:
- *GridBagConstraints.gridx, GridBagConstraints.gridy*;
- *GridBagConstraints.gridwidth, GridBagConstraints.gridheigh*;
- *GridBagConstraints.insets*;
- *GridBagConstraints.fill* – conține informații de redimensionare a componentei descrise: *GridBagConstraints.NONE*, *GridBagConstraints.HORIZONTAL*, *GridBagConstrints.VERTICAL*, *GridBagConstraints.BOTH*;
- *GridBagConstraints.anchor*.

Interogarea bazelor de cunoştinţe în limbaj natural prin tehnologia client-server

- Entitățile de bază ale tehnologiei client-server, sunt: clientul, serverul şi rețeaua.
- Conceptele de bază sunt: hostul, adresa de Internet, portul şi protocoalele.
- În continuare prezentăm o aplicație în care clientul adresează o întrebare serverului.
    1. Se lansează aplicația server (figura nr. 3). Serverul aşteaptă conectarea unui client.
    2. Se lansează aplicația client, care transmite un mesaj către server, introdus de la tastatură (figura nr. 4).
- Aplicația server primeşte mesajul de la client, îl afişează pe ecran şi îl trimite înapoi clientului, care la rândul lui afişează pe ecran acest mesaj (figura nr. 7).

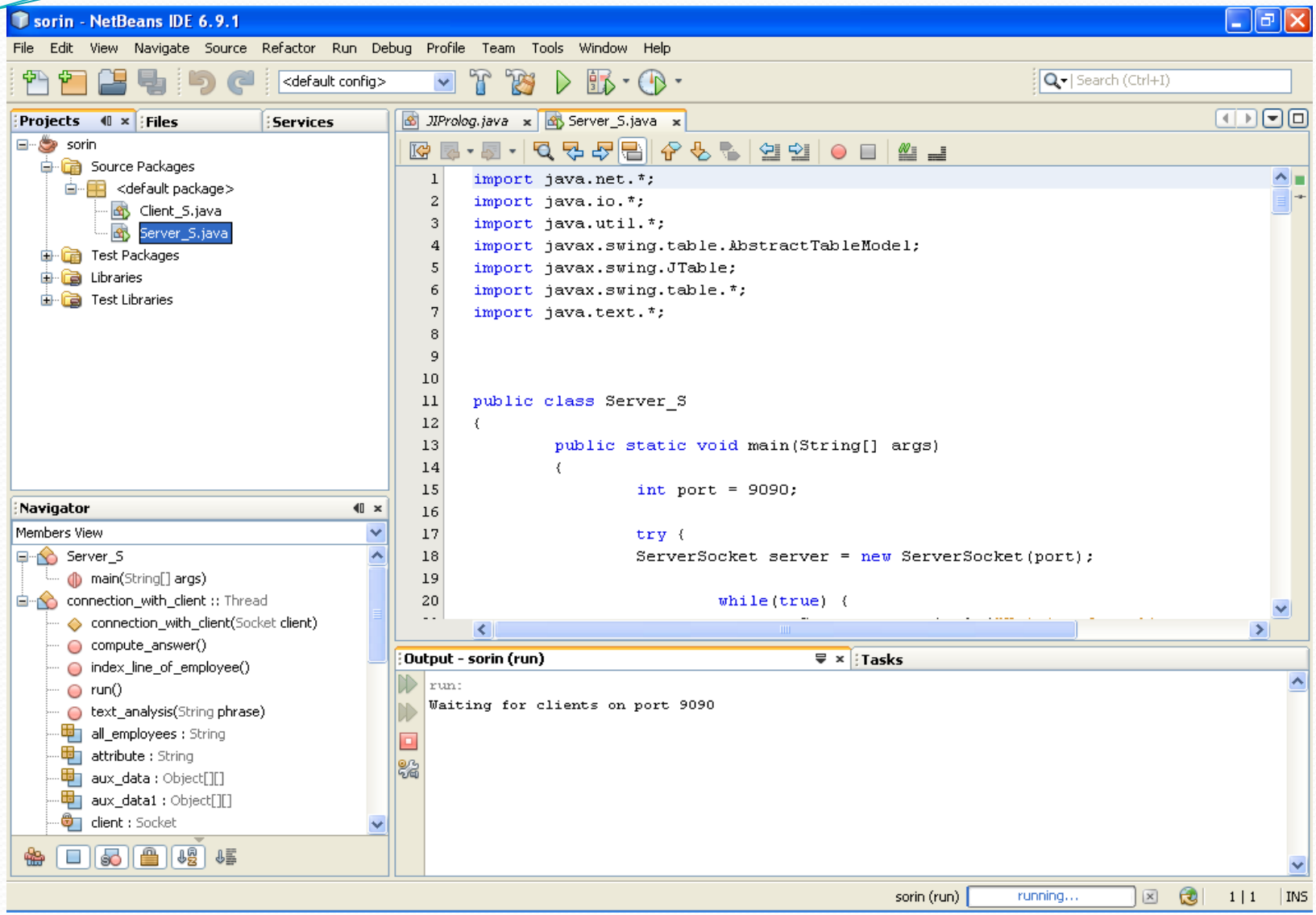

Fig. 3. Lansarea în execuție a aplicației server

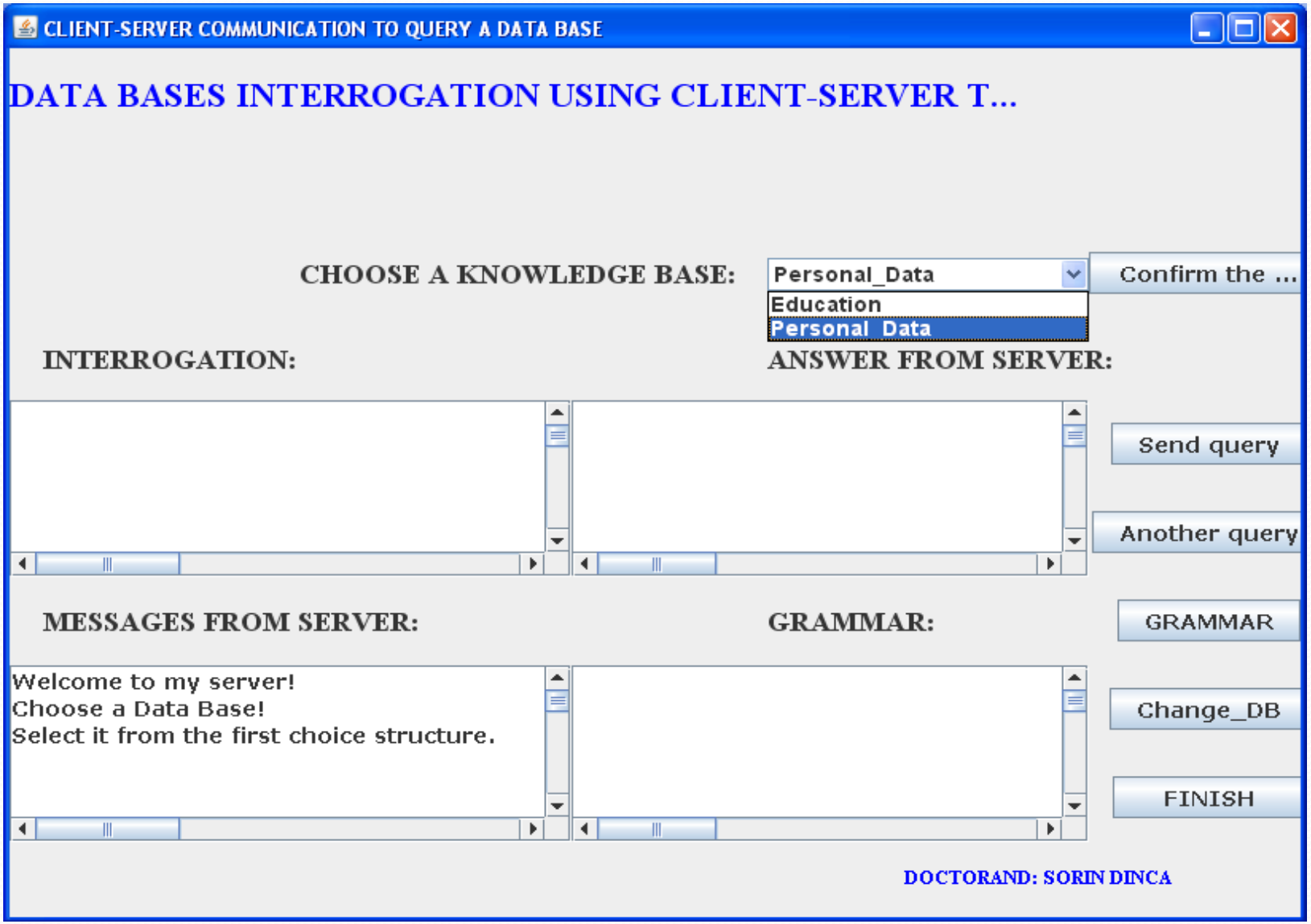

Fig. 4. Lansarea în execuţie a aplicaţiei client

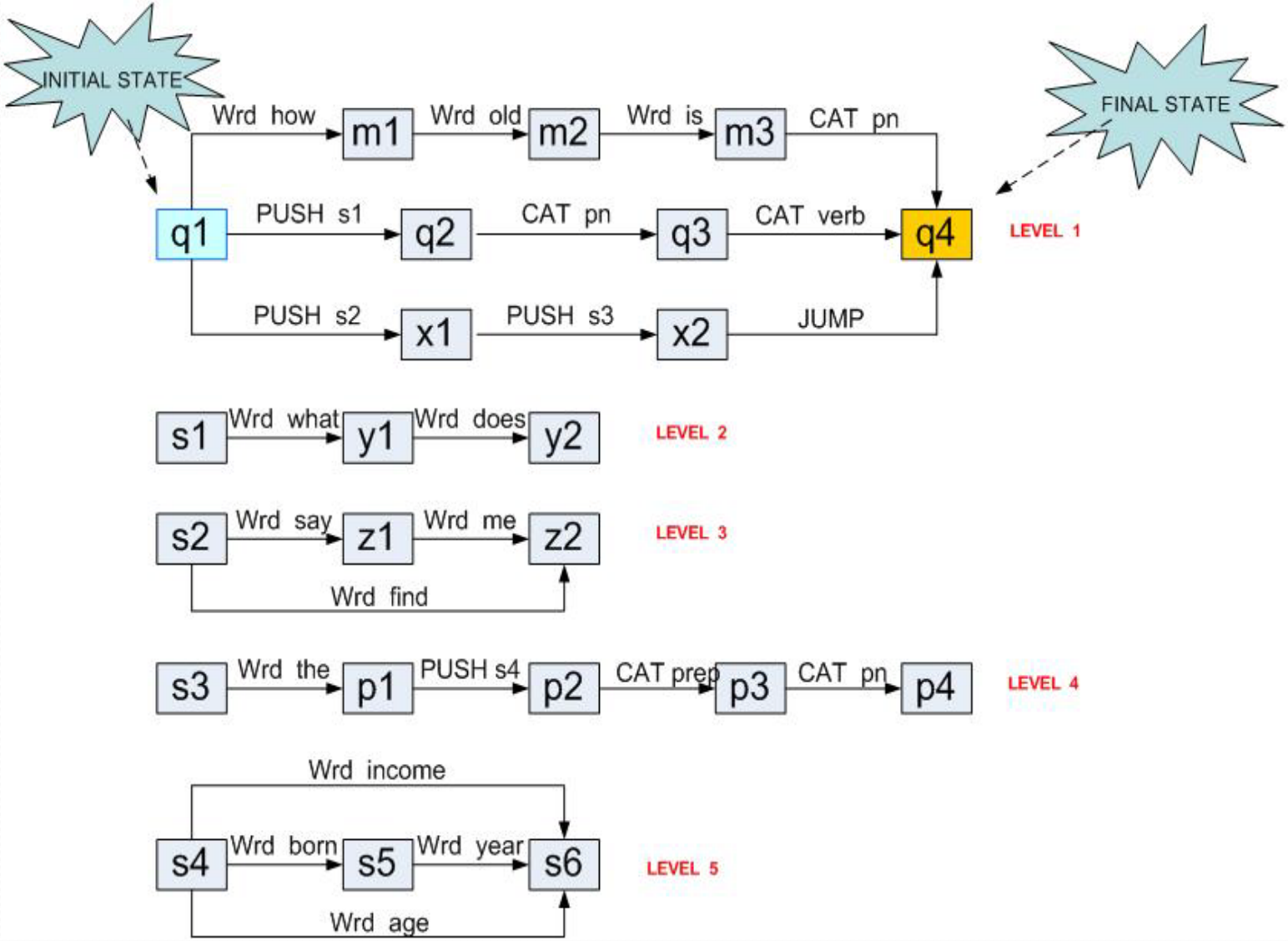

Fig. 5. Gramatica utilizată

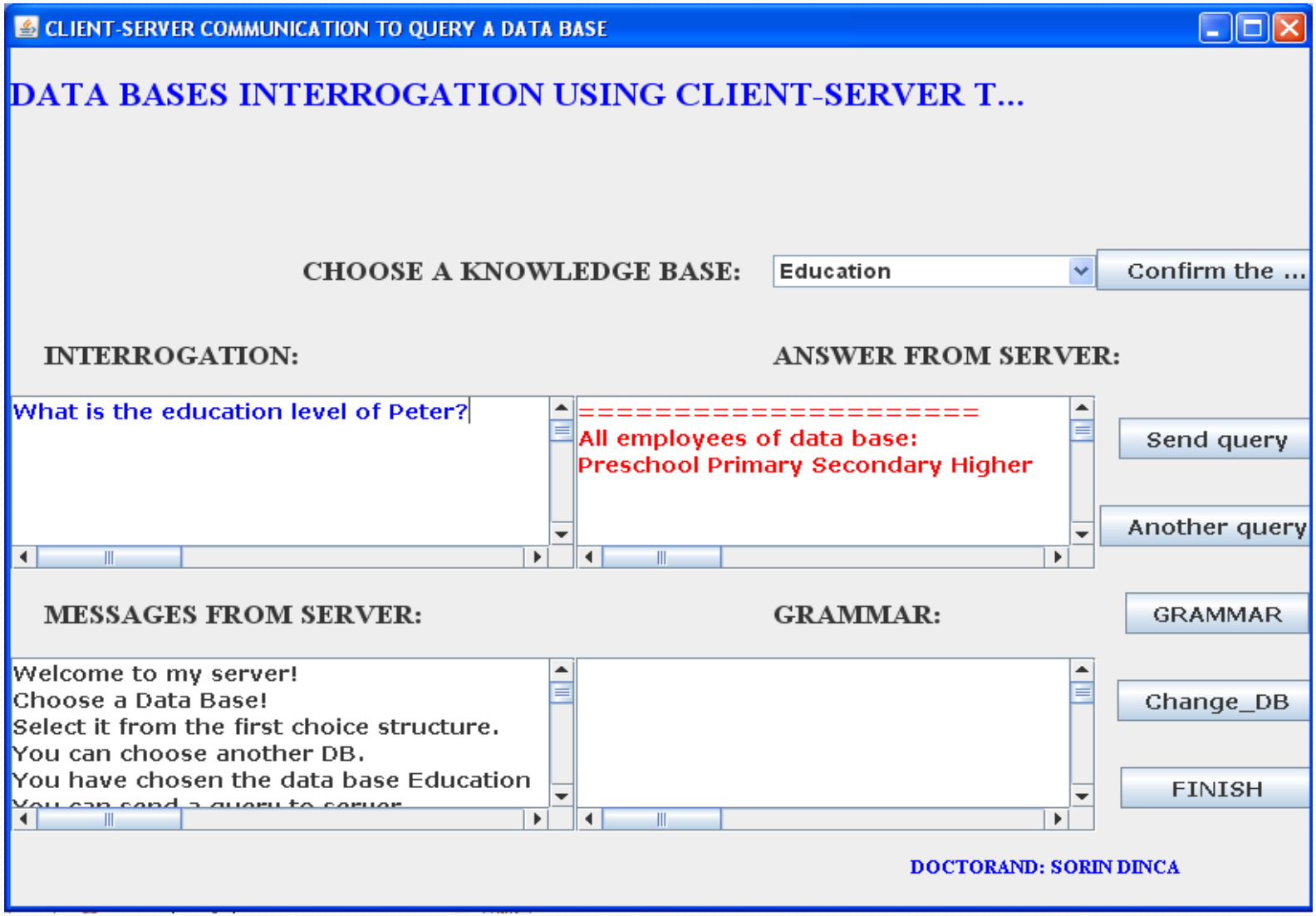

Fig. 6. Alegerea bazei de cunoştinţe

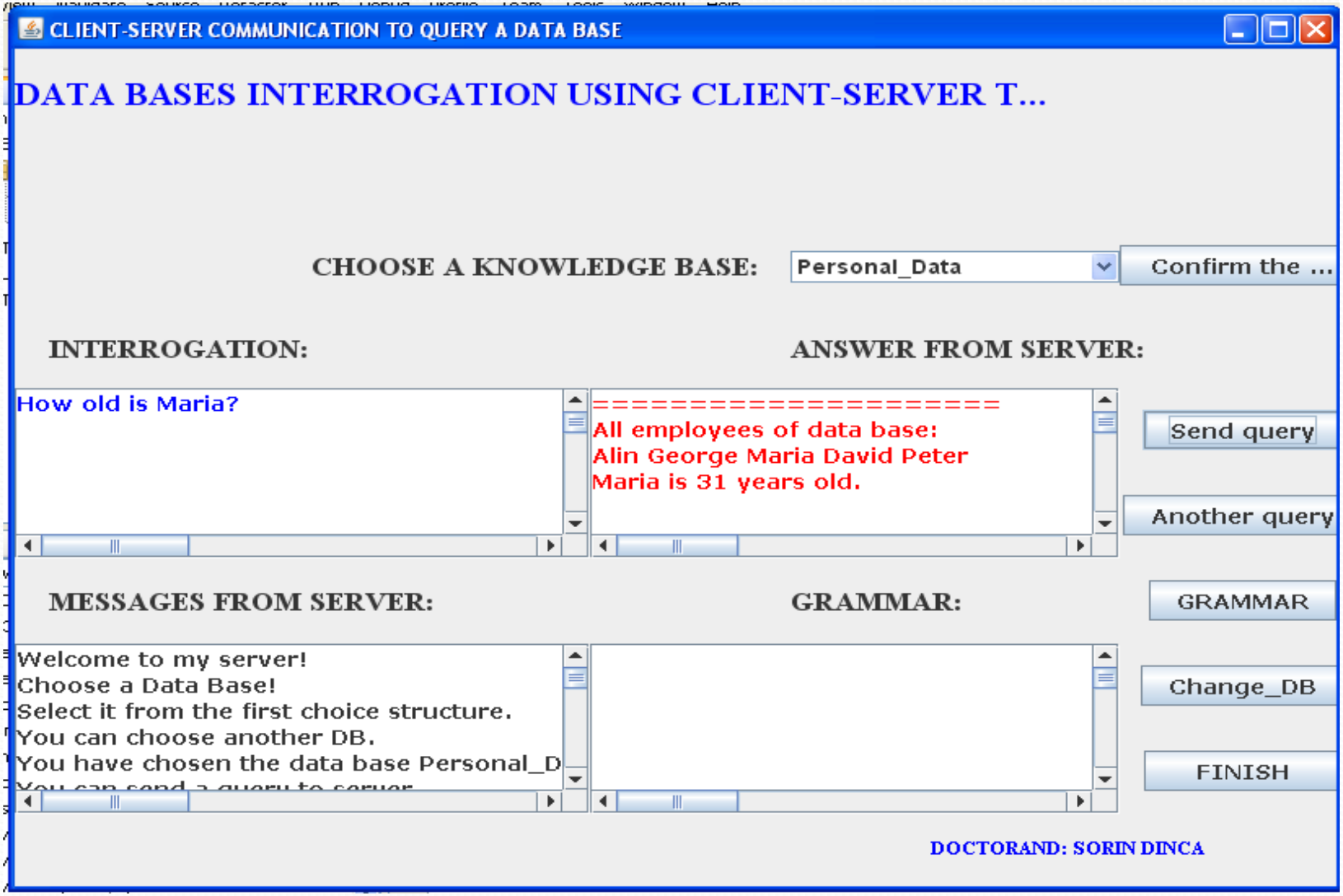

Fig. 7. Întrebarea adresată de client şi răspunsul dat de server