

# METODE INTELIGENTE DE REZOLVARE A PROBLEMELOR REALE



Laura Dioşan  
Tema 5

# Web semantic și metode de căutare

---

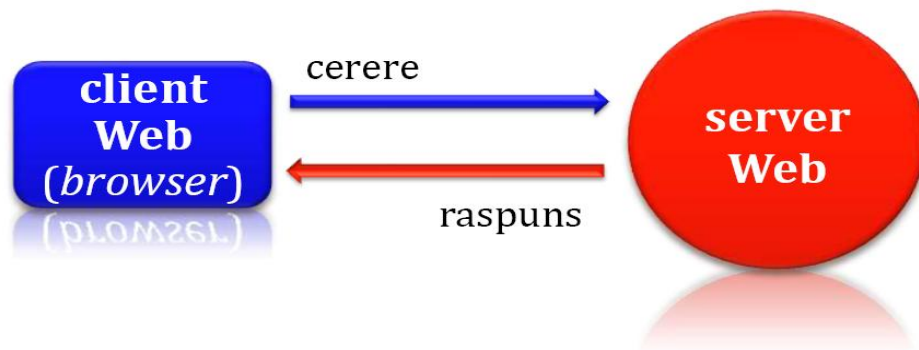
- Web - concept și evoluție
- Web semantic
- Metode de căutare

# Web - concept și evoluție

---

## □ Web (WWW)

- "pânza de păianjen mondială"
- WWW  $\neq$  Internet
- **Idee (Sir Tim Berners-Lee la CERN – 1989)**
  - integrarea unor sisteme informaționale
  - dispersate într-un mod unitar,
  - fără diferențe între sursele de date
- ***anything can link to anything***
- bazat pe:
  - modelul **client/server**
  - hypertext



# Web - concept și evoluție

## □ Web (WWW)

- "pânza de păianjen mondială"

- WWW != Internet

- **Idee (Sir Tim Berners-Lee la CERN – 1989)**

  - integrarea unor sisteme informaționale

  - dispersate într-un

  - fără diferențe între

- ***anything can link***

- bazat pe:

  - modelul **client/server**

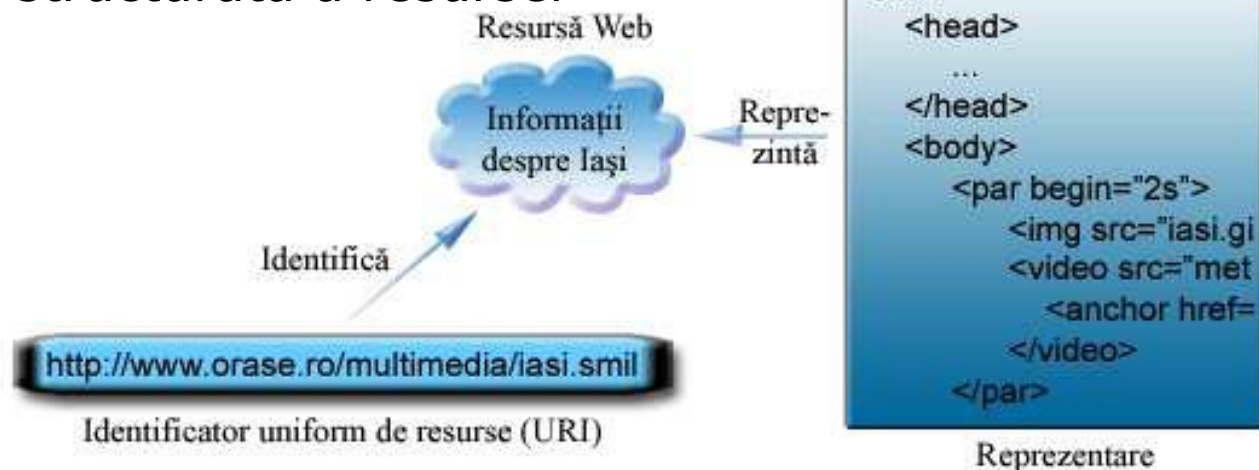
  - hypertext



# Web - concept și evoluție

## □ Arhitectură

- Resursele sunt identificate prin **adresa lor**
  - identificator uniform de resurse (URI – *Uniform Resource Identifier*)
  - <http://www.cs.ubbcluj.ro/~forest/wp/>
- Accesul la conținutul (reprezentarea) resurselor Web → **protocol**
  - HTTP – *HyperText Transfer Protocol*
- Resursele – documentele (pagini Web) – includ **<marcaje/>**
  - marcajele conțin la rândul lor URI-uri → **hipertext**
- Relațiile dintre o resursă Web, adresa ei (URI) și reprezentarea structurată a resursei



# Web - concept și evoluție

---

## □ Web 1.0

### ■ Sit Web

- sistem pe care rulează un server Web găzduind o serie de pagini (documente) WWW înrudite – ale unei organizații, companii sau persoane

### ■ Aplicație Web

#### □ Interfață

- HTML, CSS, Ajax, Flash, Silverlight, SVG, *widget-uri*,...

#### □ + Conținut (Data)

- relationale (SQL), XML, grafuri, modelare semantica (RDF)

#### □ + Program

- server: C#, Java, Perl, PHP, Ruby etc.; client: JavaScript

# Web - concept și evoluție

---

- Web 2.0 (web social)
  - WWW - platforma în care utilizatorul își controlează propriile date
  - **Participare**
    - *read/write Web*
    - colaborare, comunități, conectivitate inter-personală & între aplicații
  - **Partajare de artefacte informationale**
    - documente, fotografii, multimedia, cod-sursă etc.
  - **Inteligența colectivă**
    - editare & management colaborativ al conținutului
    - aplicații de tip *wiki*
  - **Servicii și nu pachete software**
  - **Software rulat oriunde**
  - **Mediatizare (*syndication*) Web**
    - datele privitoare la un sit Web sunt expuse liber via un **flux (*feed*)** în format
      - ex. RSS (*Really Simple Syndication*) → XML
      - ex. Atom

# Web - concept și evoluție

---

- Web 3.0 (Web-ul datelor, Web-ul semantic)
  - Cum pot fi descrise la nivelul masinii aceste *web-uri*? → *modelarea cunoștințelor*
    - o manieră de a atașa **meta-date**
      - Informații privitoare la date
    - un mod de specificare a **relațiilor dintre resurse**
      - structuri de organizare a datelor în cadrul unui sau mai multor *web-uri*
    - modelarea & procesarea cunoștințelor despre „lucruri”
      - *knowledge about things*
      - realizate sistematic, formalizat → ontologii
      - create ad-hoc, manual, de către utilizatorii obișnuiți → folksonomii
      - Implicitul → explicit
        - Java = limbaj / insulă / cafea
      - Ușor de înțeles de către oameni, dar și de către calculatoare



# Web semantic

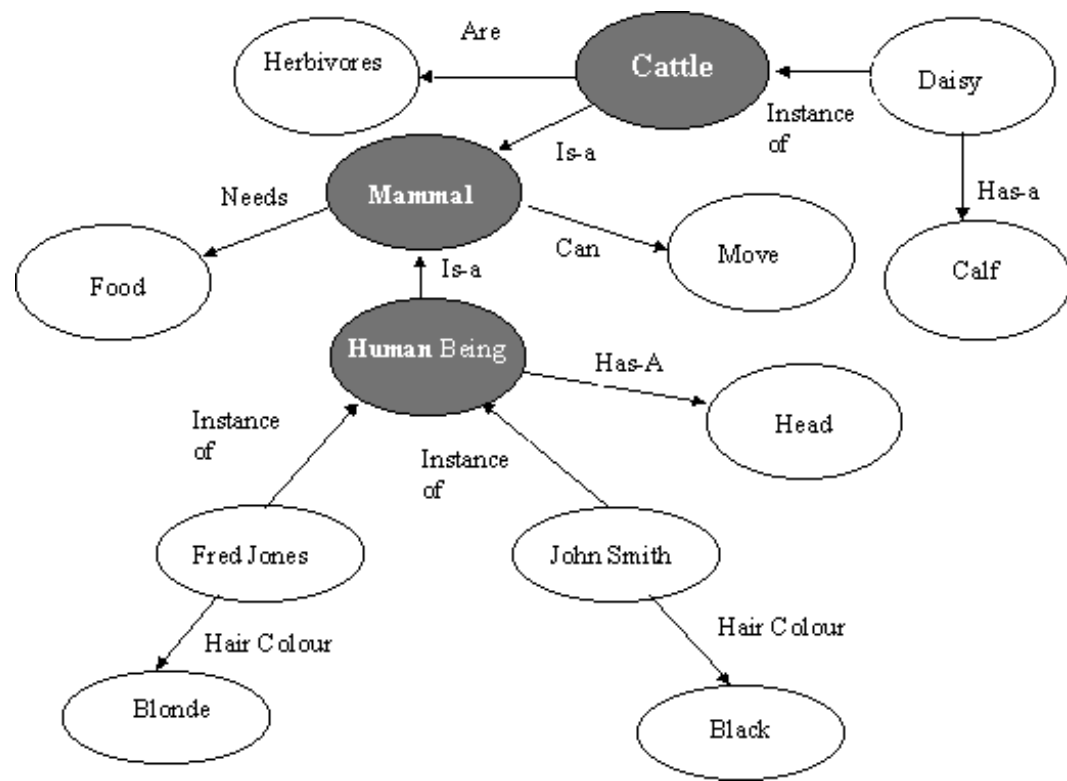
## Modelarea cunoștințelor

### □ Definirea cunoștințelor

- Meta-date → **RDF (Resource Description Framework)**

### □ Reprezentarea cunoștințelor

- Rețele semantice → graf (orientat sau neorientat)



# Web semantic

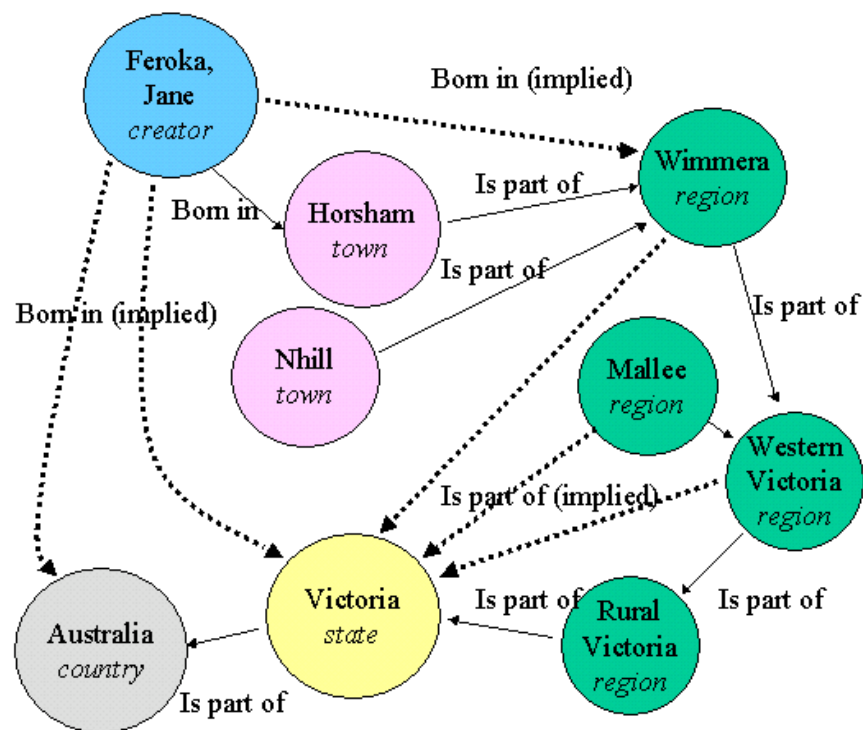
## Modelarea cunoștințelor

### □ Definirea cunoștințelor

- Meta-date → **RDF (Resource Description Framework)**

### □ Reprezentarea cunoștințelor

- Asocieri de subiecte (topic map) → Hyper-graf (relații n-are între noduri)



# Web semantic

---

## Modelarea cunoștințelor

### □ Definirea cunoștințelor

- Meta-date → **RDF (*Resource Description Framework*)**

### □ Reprezentarea cunoștințelor

- Diagrame UML

# Web semantic

---

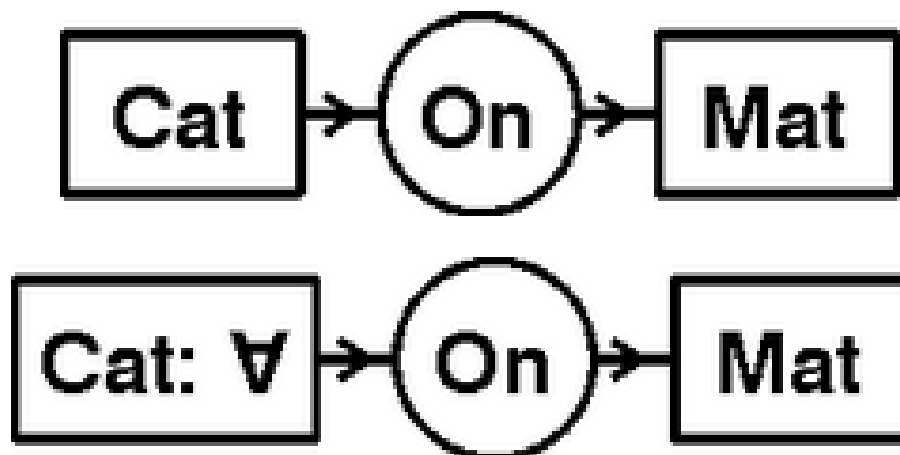
## Modelarea cunoștințelor

### □ Definirea cunoștințelor

- Meta-date → **RDF (*Resource Description Framework*)**

### □ Reprezentarea cunoștințelor

- Grafuri conceptuale → interfață grafică pentru logica de ordin I



# Web semantic

## Modelarea cunoștințelor

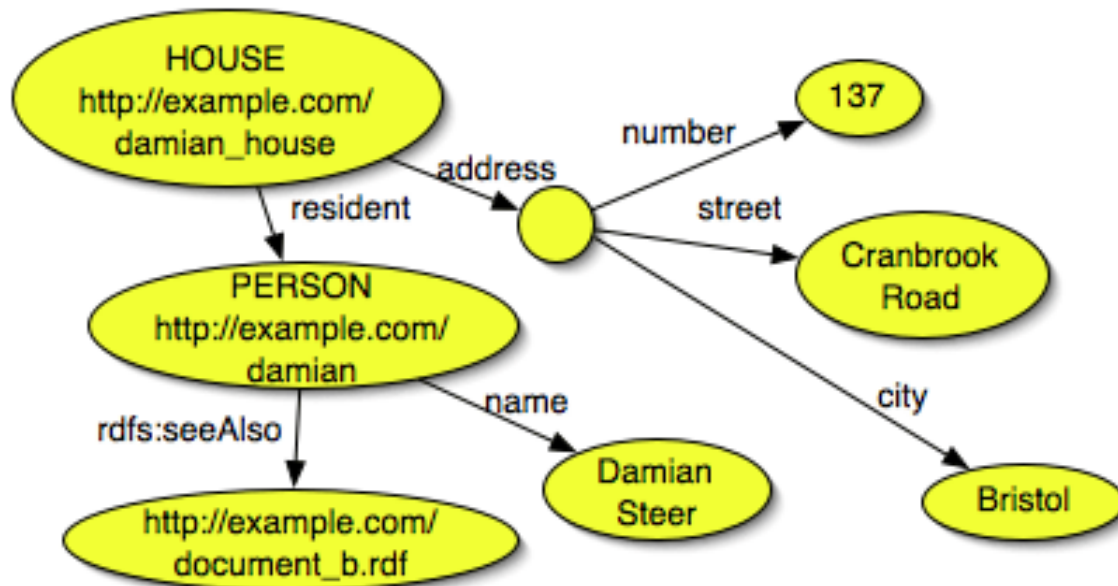
### □ Definirea cunoștințelor

- Meta-date → **RDF (Resource Description Framework)**

### □ Reprezentarea cunoștințelor

#### ■ Grafuri RDF

- ofera posibilitatea de a descrie/adnota (explicit) resursele Web
- relații ternare între elemente (subiect-predicat-complement direct)
  - Subiectul → resursa
  - Predicatul → caracteristică a resursei; leagă subiectul de complement
  - Obiect (complement) → atribut → valoare
- Ex. *Floarea are culoarea roșie*



# Web semantic

---

## Modelarea cunoștințelor

### □ Reprezentarea cunoștințelor

#### ■ taxonomii

- in termeni de clase, superclase si subclase de resurse si relatiile intre ele
- vocabular controlat (*controlled vocabulary*)
- scheme RDF

#### ■ tezaur

- *a controlled vocabulary arranged in a known order and **structured so that** equivalence, homographic, hierarchical, and associative **relationships among terms** are displayed clearly & identified by standardized relationship indicators*
- relații între resurse
  - Echivalență, omonimie, ierarhie, asociere

# Web semantic

---

## Modelarea cunoștințelor

### □ Reprezentarea cunoștințelor

#### ■ Ontologii

- Conceptualizarea unui domeniu de cunoaștere într-un format destinat a fi procesat de calculator, format modelind entități, atribute, relații și axiome
- Conține categoriile, clasele, conceptele fundamentale proprietățile conceptelor relațiile & distincțiile dintre concepte
- studiul categoriilor de lucruri (*things*) care există sau pot exista într-un domeniu de interes
- Reprezentare prin RDFS, OWL (WordNet)
- Proiectare, aliniere, fuziune

# Metode de căutare

---

- Definire
- Necesitate
- Algoritmi



# Metode de căutare

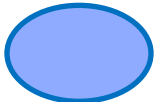

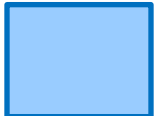



---

## □ Definire

- Identificarea celor mai bune  $k$  răspunsuri (top  $k$ ) care se potrivesc cu interogarea unui utilizator
- Rezultatul poate fi furnizat sub forma:
  - unei mulțimi
  - unei liste sortate
  - unei liste sortate și cu scoruri
- Alegerea celor  $k$  răspunsuri dintre  $N$  elemente
  - unde  $N \gg k$

# Metode de căutare

## □ Exemplu

obiect	Arie (x1)	Circularitate (x2)	Albăstreală (x3)
	0.9	0.4	0.4
	0.8	0.5	0.2
	0.6	0.1	0.3
	0.2	0.6	0.5
	0.5	0.7	0.3
	0.1	0	0

# Metode de căutare

---

## □ Necesitate

- Căutări în web și alte sarcini de regăsire/ordonare de informație utilă
  - Identificarea documentelor cu informații despre "Apps for Galaxy Smartphones"
- Căutări în depozitele cu informații multimedia
  - Identificarea imaginilor care prezintă un palmier și un apus de soare
- Căutări în depozite structurate de date pe baza preferințelor clienților
  - Identificarea apartamentelor spațioase în cartierele apropiate campusului universitar

# Metode de căutare

---

- Raportate la interogările de tip SQL
  - Relevanță graduală (nu doar bi-valentă)
  - Rezultate formate din cele mai bune exemple (nu toate exemplele care se potrivesc cu interogarea)
  - Calitatea unui exemplu este exprimată prin intermediul unui scor

# Metode de căutare

---

## □ Utilitate

- Explorează compromisul complexitate temporală vs. complexitate spațială
- Explorează domenii reale de definiție a atributelor

# Metode de căutare

---

## □ Algoritmi

- Algoritmi de potrivire (*matching*)
- Algoritmi de ordonare (*ranking*)
- Algoritmi de selecție (*top k*)

# Metode de căutare

---

## □ Algoritmi de potrivire

### ■ Potrivire exactă

- True/False

### ■ Potrivire inexactă

- distanța între 2 elemente să fie  $<$  decât un prag
- 2 elemente să aibă un nucleu comun
- 2 elemente să provină de pe aceeași ramură semantică

# Metode de căutare

## □ Algoritmi de potrivire

- Cel mai apropiat strămoș comun (*Least common ancestor – LCA*) – vers 1
  - complexitate:
    - Pre-procesare: const,
    - Căutare:  $O(\log n)$

```
LCA(root, val1, val2){
    if (root == NULL || root.getInfo() == val1 || root.getInfo() == val2)
        return -1;
    if (root.getRight() != NULL && (root.getRight().getData() == val1 ||
        root.getRight().getData() == val2 ))
        return root.getData();
    if (root.getLeft() != NULL && (root.getLeft().getData() == val1 ||
        root.getLeft().getData() == val2 ))
        return root.getData();

    if (root.getData() > val1 && root.getData() < val2)
        return root.getData();

    if (root.getData() > val1 && root.getData() > val2)
        return LCA(root.getLeft(), val1, val2)
    if (root.getData() < val1 && root.getData() < val2)
        return LCA(root.getRight(), val1, val2)
}
```



# Metode de căutare

## ❑ Algoritmi de potrivire

- Cel mai apropiat strămoș comun (*Least common ancestor - LCA*) – vers 2
  - ❑ complexitate:
    - Pre-procesare:  $O(n)$ ,
    - Căutare:  $O(\sqrt{N})$

```
LastSectionAncestor(node, F[n], n, P[n], L[n], no=sqrt(H)){  
    if (L[node] < nr) P[node] = 1;  
    else{  
        if ((L[node] % no == 0)) P[node] = F[node];  
        else P[node] = P[F[node]];  
    }  
    for each child nod of node  
        LastSectionAncestor(node, F, n, P, L, no);  
}
```

```
LCA(F[n], P[n], L[n], val1, val2){  
    while (P[val1] != P[val2]){  
        if (L[val1] > L[val2])  
            val1 = P[val1];  
        else  
            val2 = P[val2];  
    }  
    while (val1 != val2){  
        if (L[val1] > L[val2])  
            val1 = F[val1];  
        else  
            val2 = F[val2];  
    }  
    return val1;  
}
```

# Metode de căutare

---

## □ Algoritmi de ordonare

### ■ Funcții de scor (*ranking*)

- Folosite pentru a calcula scorul (importanța) unui exemplu
- Un exemplu poate avea mai multe atribute → mai multe scoruri
  - Frecvența de apariție a cuvântului "galaxy"
  - Nivelul de roșu din imagine
  - Suprafața apartamentului

### ■ Pp un exemplu $E$ cu $m$ atribute

- $E = (a_1, a_2, \dots, a_m)$
- $S(E) = g(f_1(a_1), f_2(a_2), f_3(a_3), \dots, f_m(a_m))$ 
  - $f_i$  – funcție monotonă
  - $g$  – funcție monotonă de agregare (sumă, medie, max, etc)
- Eg.
  - $g = 2 * \text{suprafața apartamentului} + 3 * \text{distanța până în campus}$

# Metode de căutare

---

## □ Algoritmi de selecție (top k)

### ■ Timpul de execuție

- Acces secvențial la exemple → iterator
- Access aleator → cf. unei chei primare
  - Mai costisitor decât accesul secvențial

### ■ Spațiul de execuție

- Câte exemple relevante se rețin la un moment dat?
  - Un anumit număr ( $k$ )?
  - Un număr dependent (liniar) de mărimea setului de date ( $N$ )?

# Metode de căutare

---

## □ Algoritmi de selecție (top k)

### ■ Abordarea naivă

#### □ Ideea de bază

- Calcularea scorului pentru fiecare exemplu
- Sortarea exemplurilor pe baza scorului
- Selectarea primelor k exemple



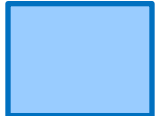



#### □ Proprietăți

- Simplă
- Complexitate liniară  $O(n)$

# Metode de căutare

## □ Algoritmi de selecție (top k)





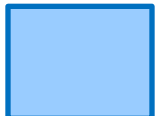

### ■ Abordarea naivă

obiect	Arie (x1)	Circularitate (x2)	Albăstreală (x3)	Scor total
	0.9	0.4	0.4	1.7
	0.8	0.5	0.2	1.5
	0.6	0.1	0.3	1
	0.2	0.6	0.5	1.3
	0.5	0.7	0.3	1.5
	0.1	0	0	0.1

# Metode de căutare

## ❑ Algoritmi de selecție (top k)

### ■ Abordarea naivă

obiect	Arie (x1)	Circularitate (x2)	Albăstreală (x3)	Scor total
	0.9	0.4	0.4	1.7
	0.8	0.5	0.2	1.5
	0.5	0.7	0.3	1.5
	0.2	0.6	0.5	1.3
	0.6	0.1	0.3	1
	0.1	0	0	0.1

# Metode de căutare

---

## □ Algoritmi de selecție (top k)

### ■ Algoritmul lui Fagin

#### □ Ideea de bază

- Precalcularea scorurilor/atribut și formarea a m liste
- Sortarea acestor m liste
- Accesarea sequentială și în paralel a acestor liste până când au fost vizitate k obiecte în fiecare listă
- Calcularea scorului pentru obiectele accesate
- Sortarea pe baza scorului și returnarea primelor k exemple

#### □ Proprietăți

- Complexitate  $O(n^{m-1/m}k^{1/m})$

# Metode de căutare

---

## □ Algoritmi de selecție (top k)

### ■ Algoritmul lui Fagin

#### □ Pp. că:

- $d \rightarrow n$  exemple cu  $m$  attribute fiecare
- $s \rightarrow$  cele  $n$  exemple sortate după fiecare atribut în parte

```
i = 0;
while (i < n) && (!stopCond){
    for(j=0; j<m; j++){
        Object o(s[i][j].index, j, s[i][j].value)
        seenObj.add(o)
    }
    no = 0;
    for each o ∈ seenObj{
        if (o.getNoFilledAtrib() == m) no++;
    }
    stopCond = (no == k);
    i++;
}
```



# Metode de căutare

## ❑ Algoritmi de selecție (top k)

### ■ Algoritmul lui Fagin

obiect	Arie (x1)
	0.9
	0.8
	0.6
	0.5
	0.2
	0.1

obiect	Circularitate (x2)
	0.7
	0.6
	0.5
	0.4
	0.1
	0


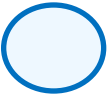

obiect	Albăstreală (x3)
	0.5
	0.4
	0.3
	0.3
	0.2
	0



















# Metode de căutare

Algoritmi de selecție (top k)

Algoritmul lui Fagin

□  $i=1$

obiect	(x1)	(x2)	(x3)
	0.9		
		0.7	
			0.5

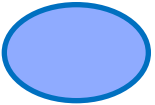
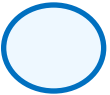


obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selecție (top k)

Algoritmul lui Fagin

□  $i=2$

obiect	(x1)	(x2)	(x3)
	0.9		0.4
		0.7	
		0.6	0.5
	0.8		


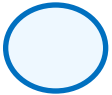


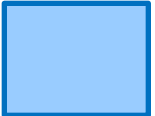
obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selecție (top k)

Algoritmul lui Fagin

□  $i=3$

obiect	(x1)	(x2)	(x3)
	0.9		0.4
		0.7	
		0.6	0.5
	0.8	0.5	
	0.6		0.3


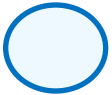


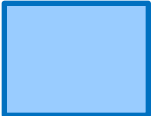
obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selecție (top k)

Algoritmul lui Fagin

□  $i=4$

obiect	(x1)	(x2)	(x3)
	0.9	0.4	0.4
	0.5	0.7	0.3
		0.6	0.5
	0.8	0.5	
	0.6		0.3

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0


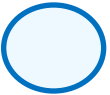


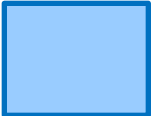
# Metode de căutare

Algoritmi de selecție (top k)

Algoritmul lui Fagin

---

□  $i=4$

obiect	(x1)	(x2)	(x3)	Global
	<b>0.9</b>	<b>0.4</b>	<b>0.4</b>	<b>1.7</b>
	<b>0.5</b>	<b>0.7</b>	<b>0.3</b>	<b>1.5</b>
		0.6	0.5	
	0.8	0.5		
	0.6		0.3	

# Metode de căutare

## □ Algoritmi de selectare (top k)

### ■ Algoritmul cu prag

#### □ Ideea de bază

Execută

Pentru fiecare exemplu **E** care a fost accesat cel puțin o dată în oricare dintre listele individuale

Accesează attributele lui E din listele în care exemplul nu a fost încă accesat

Calculează **t(E)** și actualizează lista primelor k exemple (**Y**) – dacă este necesar

Calculează  $\tau = t(\underline{f}_1, \underline{f}_2, \dots, \underline{f}_m)$ , unde  $\underline{f}_i$  este scorul ultimului exemplu accesat în lista  $L_i$

Până când  $\tau < g$  (**g** – scorul agregat cel mai mic al setului de k exemple **Y**)

#### □ Proprietăți

##### ■ Complexitate

- verifică mai puține exemple
- $k \cdot (m-1)$  accese random
- spațiu tampon pentru maxim k exemple

# Metode de căutare

## □ Algoritmi de selectare (top k)

### ■ Algoritmul cu prag

#### □ Pp.:

- $d \rightarrow n$  exemple cu  $m$  attribute fiecare
- $s \rightarrow$  cele  $n$  exemple sortate după fiecare atribut în parte

```
i = 0;
while (i < n) && (!stopCond){
    for(j=0; j<m; j++){
        Object o(s[i][j].index, j, s[i][j].value)
        for(t = 0; t < j; t++)
            o.setInfo(t, d[s[i][j].index][t]);    //random access
        for(t = j + 1; t < m; t++)
            o.setInfo(t, d[s[i][j].index][t]);    //random access
        seenObj.add(o)
    }
    sort(seenObj, Object.getAgregation()); //reverse
    theta = agregation(s[i])
    no = 0;
    for each o ∈ seenObj{
        if (o.getAgregation() > theta) no++;
    }
    stopCond = (no == k);
    i++;
}
```

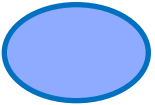
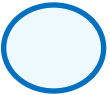





















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=1$

obiect	(x1)	(x2)	(x3)
	0.9		
		0.7	
			0.5


obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=1$

obiect	(x1)	(x2)	(x3)
	0.9	0.4	0.4
	0.5	0.7	0.3
	0.2	0.6	0.5




obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=1$

obiect	(x1)	(x2)	(x3)	Global
	0.9	0.4	0.4	1.7
	0.5	0.7	0.3	1.5
	0.2	0.6	0.5	1.3

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0

$$\Theta = 0.9 + 0.7 + 0.5 = 2.1$$


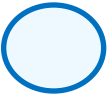


$$no = 0$$



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=2$

obiect	(x1)	(x2)	(x3)
	0.9	0.4	0.4
	0.5	0.7	0.3
	0.2	0.6	0.5
	0.8		

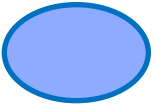
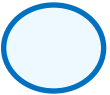

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=2$

obiect	(x1)	(x2)	(x3)
	0.9	0.4	0.4
	0.5	0.7	0.3
	0.2	0.6	0.5
	0.8	0.5	0.2





obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=2$

obiect	(x1)	(x2)	(x3)	Global
	0.9	0.4	0.4	1.7
	0.5	0.7	0.3	1.5
	0.2	0.6	0.5	1.3
	0.8	0.5	0.2	1.5

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0

$$\Theta = 0.8 + 0.6 + 0.4 = 1.8$$





$$no = 0$$



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=2$

obiect	(x1)	(x2)	(x3)	Global
	0.9	0.4	0.4	1.7
	0.5	0.7	0.3	1.5
	0.8	0.5	0.2	1.5
	0.2	0.6	0.5	1.3

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0

$$\Theta = 0.8 + 0.6 + 0.4 = 1.8$$

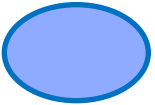
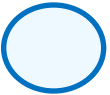


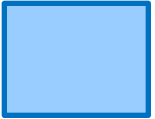
$$no = 0$$



















# Metode de căutare

## Algoritmi de selectare (top k)

### Algoritmul cu prag

□  $i=3$

obiect	(x1)	(x2)	(x3)
	0.9	0.4	0.4
	0.5	0.7	0.3
	0.2	0.6	0.5
	0.8	0.5	0.2
	0.6		0.3

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0

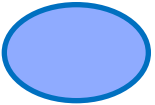
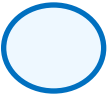


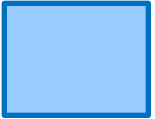




















# Metode de căutare

## Algoritmi de selectare (top k)

### Algoritmul cu prag

□  $i=3$

obiect	(x1)	(x2)	(x3)
	0.9	0.4	0.4
	0.5	0.7	0.3
	0.2	0.6	0.5
	0.8	0.5	0.2
	0.6	0.1	0.3






obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0



















# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul cu prag

□  $i=3$

obiect	(x1)	(x2)	(x3)	Global
	<b>0.9</b>	<b>0.4</b>	<b>0.4</b>	<b>1.7</b>
	<b>0.5</b>	<b>0.7</b>	<b>0.3</b>	<b>1.5</b>
	<b>0.8</b>	<b>0.5</b>	<b>0.2</b>	<b>1.5</b>
	0.2	0.6	0.5	1.3
	0.6	0.1	0.3	1.0

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0

$$\Theta = 0.6 + 0.5 + 0.3 = 1.4$$

no = 3

# Metode de căutare

---

## □ Algoritmi de selectare (top k)

### ■ Algoritmul fără acces random

#### □ Ideea de bază

Se accesează toate listele, în paralel, secvențial

După fiecare mișcare a cursorului

Se calculează scorul cel mai slab și cel mai bun al fiecărui exemplu

Se sortează obiectele vizitate pe baza celui mai slab scor

se stabilește pragul (ca prin agregarea scorurilor curente)

se oprește căutarea dacă scorul cel mai slab al celui de-al k-lea exemplu depășește pragul

Se returnează primele k obiecte

#### □ Proprietăți

- Complexitate bazată doar pe accesul secvențial

# Metode de căutare

## □ Algoritmi de selectare (top k)

### ■ Algoritmul fără acces random

#### □ Pp.:

- $d \rightarrow n$  exemple cu  $m$  attribute fiecare
- $s \rightarrow$  cele  $n$  exemple sortate după fiecare atribut în parte



















```
i = 0; limits = [];  
while (i < n) && (!stopCond){  
    for(j=0; j<m; j++){  
        Object o(s[i][j].index, j, s[i][j].value;  
        seenObj.add(o);  
        limits[j] = s[i][j].value;  
    }  
    theta = agregation(s[i])  
    for each o ∈ seenObj{  
        o.setWorst(o.getAgregation());  
        info=[];  
        for(j = 0; j < m; j++)  
            if (o.getInfo(j) == 0) info[j] = limits[j];  
            else info[j] = o.getInfo(j);  
        o.setBest(agregation(info);  
    Sort(seenObj, Object.getWorst());           //reverse  
    minTopk = seenObj[k - 1].getWorst();  
    stopCond = (theta < minTopk);  
    i++;  
}
```

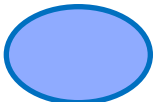
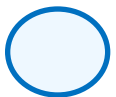

# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=1$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0
















limits	0	0	0				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9						
		0.7					
			0.5				

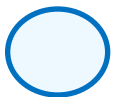

# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=1$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0
















limits	0.9	0.7	0.5				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9			0.9	2.1	2.1	0.5
		0.7		0.7	2.1		
			0.5	0.5	2.1		

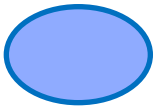
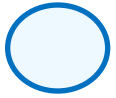

# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□ i=2

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0



















limits	0.9	0.7	0.5				
obiect	(x1)	(x2)	(x3)	worst	best	θ	topMinK
	0.9		0.4	0.9	2.1	2.1	0.5
		0.7		0.7	2.1		
		0.6	0.5	0.5	2.1		
	0.8						

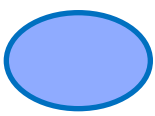
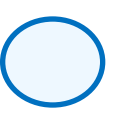


# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=2$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0

limits	0.8	0.6	0.4				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9		0.4	1.3	1.9	1.8	0.5
		0.7		0.7	1.9		
		0.6	0.5	1.1	1.9		
	0.8			0.8	1.8		





















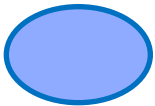
# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=2$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
	0.1		0		0
















limits	0.8	0.6	0.4				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9		0.4	1.3	1.9	1.8	1.1
		0.6	0.5	1.1	1.9		
	0.8			0.8	1.8		
		0.7		0.7	1.9		



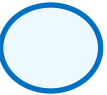
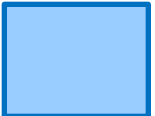
# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=3$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0
















limits	0.8	0.6	0.4				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9		0.4	1.3	1.9	1.8	1.1
		0.6	0.5	1.1	1.9		
	0.8	0.5		0.8	1.8		
		0.7		0.7	1.9		
	0.6		0.3				



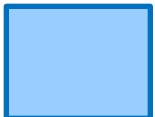
# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=3$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0
















limits	0.6	0.5	0.3				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9		0.4	1.3	1.8	1.4	1.1
		0.6	0.5	1.1	1.7		
	0.8	0.5		1.3	1.6		
		0.7		0.7	1.6		
	0.6		0.3	0.9	1.4		


# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=3$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0
















limits	0.6	0.5	0.3				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9		0.4	1.3	1.8	1.4	1.3
	0.8	0.5		1.3	1.6		
		0.6	0.5	1.1	1.7		
	0.6		0.3	0.9	1.4		
		0.7		0.7	1.6		

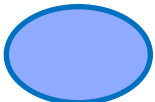



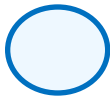
# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=4$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0
















limits	0.6	0.5	0.3				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9	0.4	0.4	1.3	1.8	1.4	1.3
	0.8	0.5		1.3	1.6		
		0.6	0.5	1.1	1.7		
	0.6		0.3	0.9	1.4		
	0.5	0.7	0.3	0.7	1.6		

# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=4$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0
















limits	0.5	0.4	0.3				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9	0.4	0.4	1.7	1.7	1.2	1.3
	0.8	0.5		1.3	1.6		
		0.6	0.5	1.1	1.6		
	0.6		0.3	0.9	1.3		
	0.5	0.7	0.3	1.5	1.5		

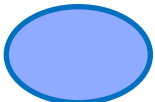
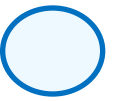


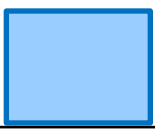
# Metode de căutare

Algoritmi de selectare (top k)

Algoritmul fără acces random

□  $i=4$

obiect	Arie (x1)	obiect	Circularitate (x2)	obiect	Albăstreală (x3)
	0.9		0.7		0.5
	0.8		0.6		0.4
	0.6		0.5		0.3
	0.5		0.4		0.3
	0.2		0.1		0.2
★	0.1	★	0	★	0

limits	0.5	0.4	0.3				
obiect	(x1)	(x2)	(x3)	worst	best	$\theta$	topMinK
	0.9	0.4	0.4	1.7	1.7	1.2	1.5
	0.5	0.7	0.3	1.5	1.5		
	0.8	0.5		1.3	1.6		
		0.6	0.5	1.1	1.6		
	0.6		0.3	0.9	1.3		