CSE 344 - Introduction to Database Management

Datalog, RC, Cost Estimation

Datalog Preview

Datalog, introduction

- A subset of Prolog
- Using non-recursive datalog with negation for the purposes of this class
- Not implementing datalog (Use DLV if you’re curious)
- Used in big data applications (Google page rank algorithm)

Datalog, a brief overview

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

- A datalog *rule:* Q1(y) :- Movie(x,y,1940)
- Basically a query
- A datalog *fact:* Actor(7920, ‘Tom’, ‘Hanks’)
- Basically a tuple
- Like RC, uses an *unnamed perspective*, meaning attributes are defined by position rather than name

A more complex rule

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

- Q2(b) :- Actor(z, ‘Tom’, ‘Hanks’), Casts(z, a), Movie(a, b, 1995)

A more complex rule

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

- Q2(b) :- Actor(z, ‘Tom’, ‘Hanks’), Casts(z, a), Movie(a, b, 1995)
- Names of movies released in 1995 that Tom Hanks was cast in

SQL Alternative:

```
SELECT m.name
FROM Actor a, Casts c, Movie m
WHERE a.id = c.pid
AND c.mid = m.id
AND a.fname = ‘Tom’
AND a.lname = ‘Hanks’
AND year =1995
```

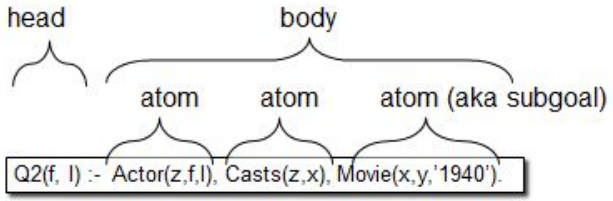
Facts:
Q2(‘Apollo 13’)
Q2(‘Toy Story’)

Datalog continued

Actor(id, fname, lname)
Casts(pid, mid)
Movie(id, name, year)

- *Program:* A collection of rules
- *External database relations:* Input relations (Actors, Casts, Movies)
- *Internal database relations:* Output relations (Q1, Q2, Q3, B1, B2, Q4)

Datalog: Terminology



f, l = head variables
x,y,z = existential variables

Safety

- Two unsafe queries:
  - U1(x,y) :- Movie(x,z,1994)
  - U2(x) :- Movie(x,z, 1994), not Casts(u,x)
- No recursion, for example:
  - T(x,y) :- E(x,y)
  - T(x,z) :- E(x,y), T(y,z)

Example problem

- Creating a datalog program that uses negation

Consider the following database schema:
Neighbors(name1, name2, duration)
Colleagues(name1, name2, duration)

Write datalog query that returns all neighbors who do not have any colleagues in common:

Example problem

- Creating a datalog program that uses negation

  Consider the following database schema:
  Neighbors(name1, name2, duration)
  Colleagues(name1, name2, duration)

  Write datalog query that returns all neighbors who do not have any colleagues in common:

  NonAnswers(n1, n2) :- Neighbors(n1, n2, -), Colleagues(n1, c, -), Colleagues(n2, c, -)

  A(n1, n2) :- Neighbors(n1, n2, -), NOT NonAnswers(n1, n2)

Cost Estimation Revisited

Cost Parameters

- Cost = I/O + CPU + Network BW
  - We will focus on I/O
- Parameters:
  - **B(R)** = # of blocks (i.e., pages) for relation R
  - **T(R)** = # of tuples in relation R
  - **V(R, a)** = # of distinct values of attribute a
    - When **a** is a key, **V(R,a) = T(R)**
    - When **a** is not a key, **V(R,a)** can be anything < **T(R)**
- Where do these values come from?
  - DBMS collects statistics about data on disk

Estimating Cost

We have 3 relations:

```
Student(sid, name, age, addr)  Book(bid,
title, author)  Checkout(sid, bid, date)
```

We want to run this query:

```
SELECT S.name
FROM Student S, Book B, Checkout C  WHERE S.sid =
C.sid
AND B.bid = C.bid
AND B.author = 'Vladimir Putin'  AND S.age > 11
AND S.age < 20
```

S(sid, name, age, addr)
B(bid, title, author)
C(sid, bid, date)

Assumptions

Student: S, Book:B, Checkout: C

Sid, bid foreign key in C referencing S and B resp.

Clustered index on C(bid, sid)

There are 10,000 Student records stored on 1,000 pages.

There are 50,000 Book records stored on 5,000 pages.

There are 300,000 Checkout records stored on 15,000 pages.

There are 8,000 unique students who have an entry in Checkout

There are 10,000 unique books that are referenced in Checkout

There are 500 different authors.

8 <= student age <= 23

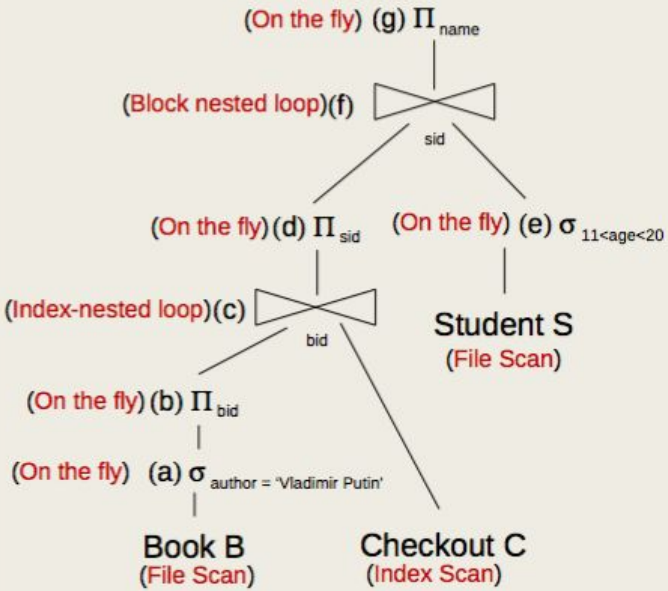
V(B, author) = 500
V(C, sid) = 8000
V(C, bid) = 10000
8 <= age <= 23

T(S)=10,000
T(B)=50,000
T(C)=300,000

B(S)=1,000
B(B)=5,000
B(C)=15,000

S(sid, name, age, addr)
B(bid, title, author)
C(sid, bid, date)

Selectivity



- (a) $\sigma_{\text{author = 'Vladimir Putin'}}$
- (c) Join predicate bid
- (e) $\sigma_{11<age<20}$
- (f) Join predicate sid

- *(a) 1 / V(B, author)*
  = 1 / 500
- *(c) 1 / max(V(B, bid), V(C, bid))*
  = 1 / max(50000, 10000)
  = 1 / 50000
- *(e) (# ages covered) / (# possible ages)*
  = 8 / 16
  = 1 / 2
- *(f) 1 / max(V(C, sid), V(S, sid))*
  = 1 / max(8000, 10000)
  = 1 / 10000

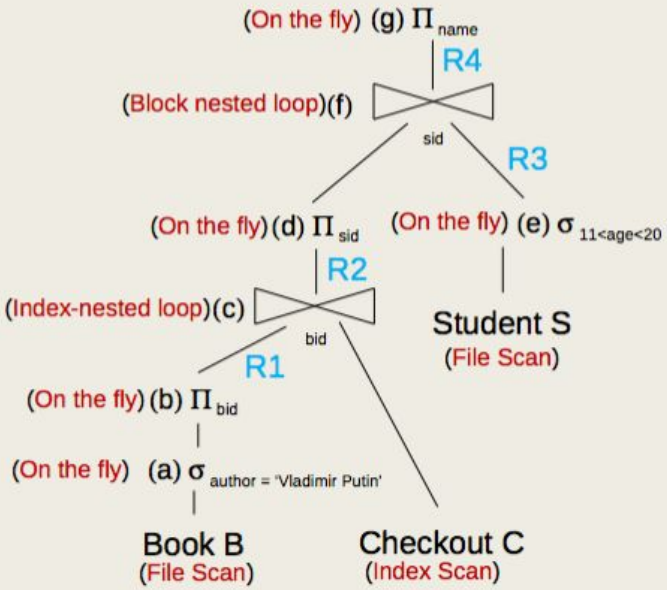
V(B, author) = 500
V(C, sid) = 8000
V(C, bid) = 10000
8 <= age <= 23

T(S)=10,000
T(B)=50,000
T(C)=300,000

B(S)=1,000
B(B)=5,000
B(C)=15,000

S(sid, name, age, addr)
B(bid, title, author)
C(sid, bid, date)

Cardinality



- T(R1) = ?
- T(R2) = ?
- T(R3) = ?
- T(R4) = ?

- T(R1) = T(B) / 500
  = 100
- T(R2) = T(R1) * T(C) / 50000
  = 100 * 300000 / 50000
  = 600
- T(R3) = T(S) / 2
  = 10000 / 2
  = 5000
- T(R4) = T(R2) * T(R3) / 10000
  = 600 * 5000 / 10000
  = 300

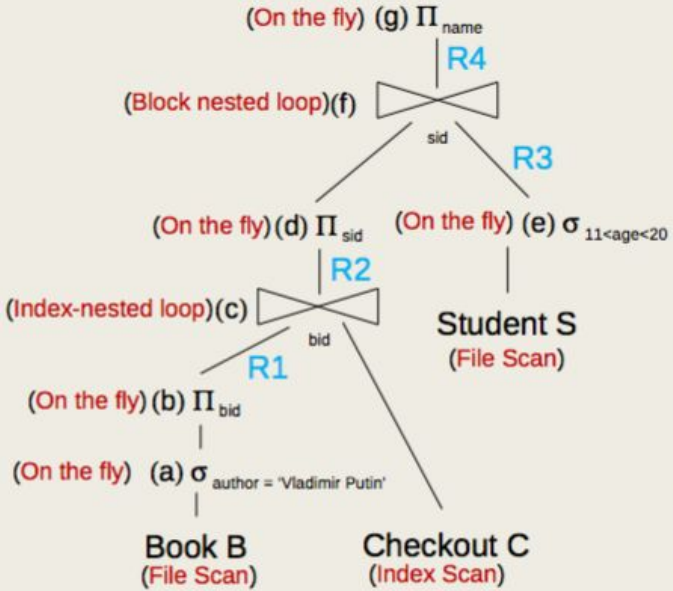
V(B, author) = 500
V(C, sid) = 8000
V(C, bid) = 10000
8 <= age <= 23

T(S)=10,000
T(B)=50,000
T(C)=300,000

B(S)=1,000
B(B)=5,000
B(C)=15,000

S(sid, name, age, addr)
B(bid, title, author)
C(sid, bid, date)

Cost



- Data not sorted in any way
- Relations can fit in memory
- Compute the cost of each step (a) through (g)

- (a) B(B) = 5000
- (b) 0
- (c) T(R1) * B(C) / V(C, bid)
  = 100 * 15000 / 10000 = 150
- (d) 0
- (e) B(S) = 1000
- (f) 0
- (g) 0
- Total: 6150

RC/Datalog Worksheet