CPSC 304 – Administrative notes November 8 and 14, 2024

- Project:
  - Make sure you have your end-to-end project tech stack working by the end of this week! I mean it!
  - Milestone 4: Project implementation – due November 29
    - You cannot change your code after this point!
  - Milestone 5: Group demo – week of December 2
  - Milestone 6: Individual Assessment – Due November 29
- Tutorials: basically project group work time/office hours
- Final exam: December 16 at 12pm! Osborne A

A bit on common project woes right now

- Tuples not showing up in the database/your application?
  → add “Commit”
- Error on table space?
  → from sqlplus:
  SELECT table_name
  From user_tables;
- Too many sessions?
  → Kill your sessions

Now where were we...

- We had been discussing Datalog
- You can read a Datalog rule of head:-body as "if I know the body, then I know the head“
- But what happens if you have multiple Datalog rules with the same name in the head?

Multiple Datalog Rules

Product (pid, name, price, category, maker-cid)
Purchase (buyer-sin, seller-sin, store, pid)
Company (cid, name, stock price, country)
Person(sin, name, phone number, city)

- Find names of people that are either buyers or sellers:

```
A(N) :- Person(S,N,A,B), Purchase(S,C,D,E)
A(N) :- Person(S,N,A,B), Purchase(C,S,D,E)
```

- Multiple rules of the same name correspond to union
- Variable names in the different rules can be the same or different

Exercise part 3

Product (pid, name, price, category, maker-cid)
Purchase (buyer-sin, seller-sin, store, pid)
Company (cid, name, stock price, country)
Person(sin, name, phone number, city)

Ex #4: Find sins of people who bought stuff from a person named Joe or bought products from a company whose stock prices is more than $50.

```
Ans(Sin):-Purchase(Sin,Ssin,_,_), Person(Ssin,"Joe",_,_)
Ans(Sin):-Purchase(Sin,_,_,Pid), Product(Pid,_,_,_,Cid),
                    Company(Cid,_,Sp,_), Sp > 50
```

Great! But surely there's more...

Our ongoing schema:

Product (pid, name, price, category, maker-cid)

Purchase (buyer-sin, seller-sin, store, pid)

Company (cid, name, stock price, country)

Person(sin, name, phone number, city)

Write a query in relational algebra to find SINs of people who have not made a purchase

$\pi_{sin}(Person) - \pi_{buyer\text{-}sin}(Purchase)$

Negation

Find people who live in Vancouver but have not bought anything at “The Bay”

```
VancouverAntiBay(buyer,seller,product,store) :-
    Person(buyer, name, phone,“Vancouver”),
    Purchase(buyer, seller, store, product),
   not  Purchase(buyer, seller, “The Bay”, product)
```

The NOT in Datalog means “there exists no”

You may also see “NOT” written as “¬ ”

Exercise part 4

Product (pid, name, price, category, maker-cid)
Purchase (buyer-sin, seller-sin, store, pid)
Company (cid, name, stock price, country)
Person(sin, name, phone number, city)

Ex #5: Find the sins of people who are not named 'Joe'

Ans(s):- Person(s,n,p,c), NOT Person(s, "Joe", p, c)

Note that we can't use anonymous variables here or the variables in the "Joe" person will not be "safe".

Rule safety

- Every variable in the head of a rule must also appear in the body.

PriceParts (Part, Price) :- Assembly(Part, Subpart, Qty) , Qty> 2.

Can generate infinite new facts (what is price)?

- Every variable must appear in a relation

Ans(Id) :- Product(Id,Name,Price,Category,Cid), Id < Stock_price

What is the value of stock_price?

- Every variable in the head of the rule must appear in some positive relation occurrence in the body and every variable in negated form must appear positively in the body

Ans(Sin):- NOT Person(Sin, 'Joe', Ph, City)

Sin, Ph, and City are unsafe

Defining Queries for reuse: Views

VancouverAntiBay(Buyer,Seller,Product,Store) :-

Person(Buyer, Bname, Phone, “Vancouver”),

Purchase(Buyer, Seller, Store, Product),

not Purchase(Buyer, Seller, “The Bay”, Product)

Ans6(Buyer) :- VancouverAntiBay(Buyer, Joe-SIN, Pro, Store),
Person(Joe-SIN, “Joe”, J-phone, pid)

Ans6(Buyer) :- VancouverAntiBay(Buyer, Sell, Prod, Store),
Product(Prod, Pname, Price, Cat, Maker)
Company(Maker, Sp, Country), Sp > 50.

What is returned by Ans6?

Buyers from Vancouver that have never purchased anything from “The Bay” that have either bought from Joe or products that are from companies with SP> 50

Clicker exercise

Flight

orig	dest
YVR	SEA
YVR	PIT
YVR	RDU
SEA	PIT
PIT	RDU
RDU	ITH

- Consider Flight(orig,dest):
- Compute Indirect_only(orig,dest) defined by:

  Twohops (Orig,Final_dest):- Flight(Orig,Mid),
  Flight(Mid,Final_dest)

  Indirect_only(orig,dest):-Twohops(orig,dest),
  NOT Flight(orig,dest)
- Which of the following tuples are in Indirect_only(orig,dest)?

A. (YVR,RDU) In Twohops(YVR,PIT)(PIT,RDU) and Flight(YVR,RDU)

B. (YVR,ITH) In Indirect_only, so correct

C. (RDU,ITH) In Flight

D. All of the above

E. None of the above

Clicker Question

Given the following schema:

Product (pid, name, price, category, maker-cid)
Purchase (buyer-sin, seller-sin, store, pid)
Company (cid, name, stock price, country)
Person(sin, name, phone number, city)

And the following query:

"Find the phone numbers of all customers who bought a computer product from a Canadian company that cost $100"

What is the proper translation into Datalog?

A. Ans(PN):- Purchase(B,_,_,P), Product(P,_,_,'computer',C),
Company(C, _, _, 'Canada'), Person(B,_,PN,_,_), price=100.

B. Ans(PN):- Purchase(B,_,_,P), Product(P,_,100, 'computer',C),
Company(C, _, _, 'Canada'), Person(B,_,PN,_,_).

C. Ans(PN):- Purchase(B,_,_,P), Product(P,_,100, 'computer',C),
Company(C, _, _, country), Person(B,_,_,_,_), country='Canada'.

D. All are correct

E. None of the above

Clicker Question

Given the following schema:

Product (pid, name, price, category, maker-cid)
Purchase (buyer-sin, seller-sin, store, pid)
Company (cid, name, stock price, country)
Person(sin, name, phone number, city)

A – price not in any atoms
C – PN not in any atoms

And the following query:

"Find the phone numbers of all customers who bought a computer product from a Canadian company that cost $100"

What is the proper translation into Datalog?

A. Ans(PN):- Purchase(B,_,_,P), Product(P,_,_,'computer',C),
Company(C, _, _, 'Canada'), Person(B,_,PN,_,_),price=100.

B. Ans(PN):- Purchase(B,_,_,P), Product(P,_,100, 'computer',C),
Company(C, _, _, 'Canada'), Person(B,_,PN,_,_).

B is correct

C. Ans(PN):- Purchase(B,_,_,P), Product(P,_,100, 'computer',C),
Company(C, _, _, country), Person(B,_,_,_,_), country='Canada'.

D. All are correct

E. None of the above

Division in Datalog

Assume schema

Cust(cid,cname,rating,salary)
Order(iid,cid,day,qty)

Query: find items (iid) that are ordered by every customer

First: write the query in Relational Algebra

$\pi_{iid,cid}(order)/\pi_{cid}(cust)$

Reminder: building up division subtract off disqualified answers in RA

$A=R$

Sno	Pno
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

$B2 = S$

Pno
P2
P4

$\pi_X(R)$

Sno
S1
S2
S3
S4

All possible values given R

$\pi_X(R) \times S$

Sno	Pno
S1	P2
S1	P4
S2	P2
S2	P4
S3	P2
S3	P4
S4	P2
S4	P4

$\pi_X(R) \times S - R$

Sno	Pno
S2	P4
S3	P4

Missing to have $\pi_X(R)$

Values needed for $\pi_X(R)$

$$\pi_X(R) - \pi_X(\pi_X(R) \times S - R) = A/B2$$

A/B2 =

Sno
S1
S4

Answers not disqualified

What does this look like in Datalog?

```
Witness(I,C):-Order(I,C,_,_)
Bad(I)       :- Cust(_,C,_,_), Order(I,_,_,_),  NOT Witness(I,C)
Good(I)      :- Order(I,C, _, _), NOT Bad(I)
```

Technically, you don't need witness, can just use another "order". It's just cleaner to do it this way.

- Witness finds all items that have been ordered
- Bad finds all items that have not been ordered by some customer
- Good: finds all items for that have not been not ordered by some customer (i.e., all items that have been ordered by all customers)

More generally, in the simplest case

Relations: A(A1, B1, …), B(B1, …)

Witness(a1,b1):-A(a1,b1,…)

Bad(a1) :- A(a1,…), B(b1,…), NOT Witness(a1,b1)

Good(a1) :- A(a1,…), NOT Bad(a1)

Returning to our previous example

A(Sno, Pno)

Sno	Pno
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B(Pno)

Pno
P2
P4

Witness(a1,b1):-
A(a1, b1)

a1	b1
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

A/B =

a1
S1
S4

Bad-no-π(a1,b1):-
A(a1,…), B(b1,…)
NOT Witness(a1,b1)

a1	b1
S2	P4
S3	P4

Bad(a1):-
A(a1,…), B(b1,…)
NOT Witness(a1,b1)

Bad

a1
S2
S3

Good(a1):-A(a1,…),
NOT Bad(a1)

Taking it to the next level

Say you're planning a beach vacation

And you wanted to find if it's possible to get from YVR to OGG (that's on Maui)

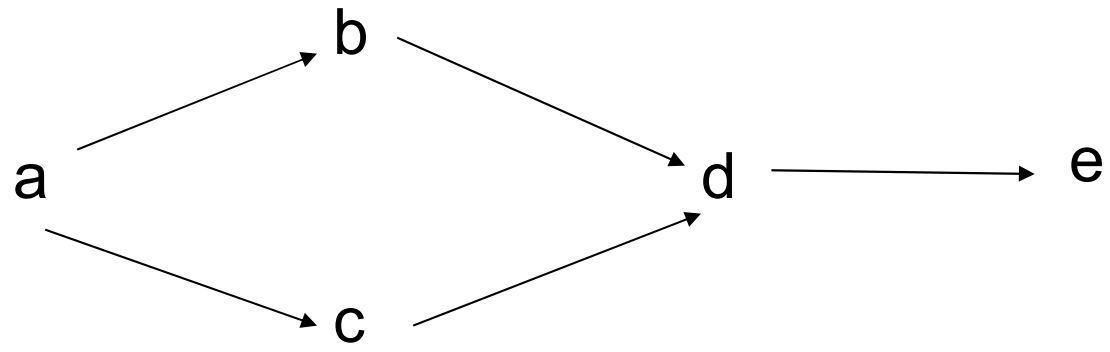
Your available information:

Flight(airline,num,origin,destination)

Now what?

A more general Example: Transitive Closure

Suppose we represent a graph w/ relation *Edge(X,Y):*
Edge(a,b), Edge (a,c), Edge(b,d), Edge(c,d), Edge(d,e)



How can I express the query: *Find all paths*

```
Path(X, Y)   :-   Edge(X, Y).
Path(X, Y)   :-   Path(X, Z),  Path(Z, Y).
```

Evaluating Recursive Queries

Path(X, Y) :- Edge(X, Y).
Path(X, Y) :- Path(X, Z), Path(Z, Y).

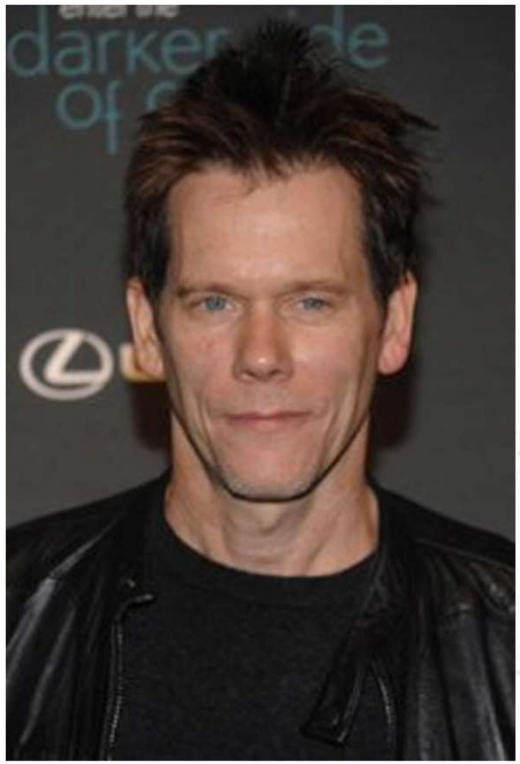
Semantics: evaluate the rules until a *fixed point:*

Iteration #0: Edge: {(a,b), (a,c), (b,d), (c,d), (d,e)}
Path: {}
Iteration #1: Path: {(a,b), (a,c), (b,d), (c,d), (d,e)}
Iteration #2: Path gets the new tuples: (a,d), (b,e), (c,e)
Path: {(a,b), (a,c), (b,d), (c,d), (d,e), (a, d), (b,e), (c, e)}
Iteration #3: Path gets the new tuple: (a,e)
Path: {(a,b), (a,c), (b,d), (c,d), (d,e), (a, d), (b,e), (c, e), (a,e)}
Iteration #4: Nothing changes → Stop.

Note: # of iterations depends on the data. Cannot be anticipated by only looking at the query!

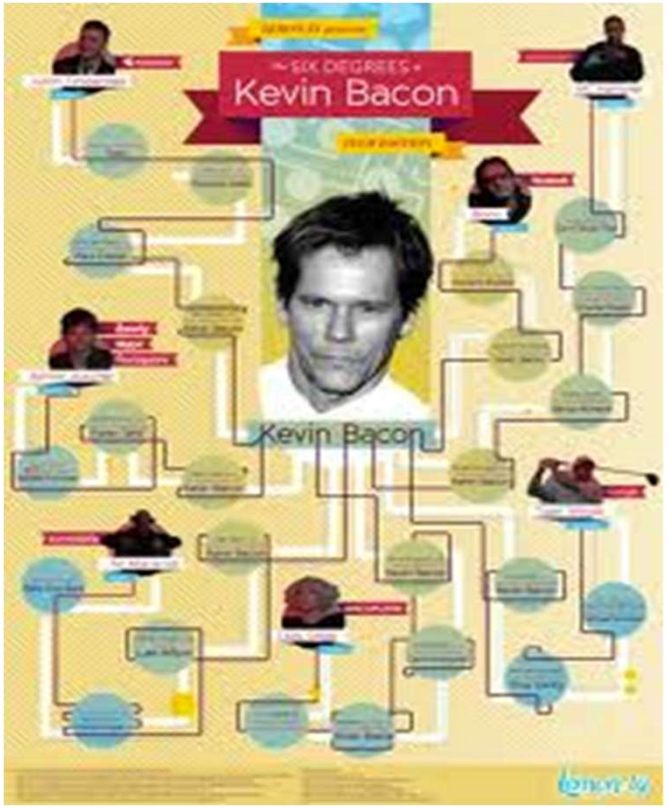
A fun Example

Kevin Bacon



6 degrees of separation

6 degrees of Kevin Bacon



More examples

- Given:
  Movie(id, title)
  Actor(id, name)
  Role(movie-id, actor-id, character)
- Find names of actors who have “Bacon numbers” (assume there’s only one “Kevin Bacon”)

```
ConnectedStars(Aid,Bid):-Role(Mid,Aid,_), Role(Mid,Bid,_)
ConnectedStars(Aid,Bid):- ConnectedStars(Aid,Cid),
                          ConectedStars(Cid,Bid)
Bacon_N(B):-Actor(Aid, “Kevin Bacon”), ConnectedStars(Aid,Bid),
            Actor(Bid,B)
```

Not tested on exam

Recursive SQL? Sometimes...

Given: Assembly(Part, Subpart, Quantity)
Find: all of the components required for a trike

Datalog:

```
Comp(Part, Subpt) :- Assembly(Part, Subpt, Qty).
Comp(Part, Subpt) :- Assembly(Part, Part2, Qty), Comp(Part2, Subpt).
```

SQL:

```
WITH RECURSIVE Comp(Part, Subpt) AS
   (SELECT A1.Part, A1.Subpt FROM Assembly A1)
  UNION
   (SELECT A2.Part, C1.Subpt
   FROM Assembly A2, Comp C1
   WHERE A2.Subpt=C1.Part)

SELECT Subpart FROM Comp C2
WHERE Part = 'trike'
```

On the off chance that you have the book… Skip the stuff on Magic Sets

- That's Datalog
- It's simple
- It's based on logic
- It's easy to see the join patterns (especially with anonymous variables)

Learning Goals Revisited

- Given a set of tuples (an input relation) and rules, compute the output relation for a Datalog program.
- Write Datalog programs to query an input relation.
- Explain why we want to extend query languages with recursive queries. Provide good examples of such queries.
- Explain the importance of safe queries, and what makes a Datalog query safe.

More practice

- Do Datalog in class exercise (for credit)
- I've posted an old Datalog tutorial and updated the link to the software. I'm sure that it at least basically works, but we won't have an explicit time for this in tutorial and it may be buggy. Should be better than nothing, though!