A Survey of Deductive Databases

Raghu Ramakrishnan and Jeffrey D. Ullman

CS 848, Fall 2016
University of Waterloo
Presented by: Siddhartha Sahu

Overview

- Relational Databases
- Deductive Databases
- Datalog
- Example Queries
- Query Execution
- Conclusion and Discussion

Relational Databases

Relational Databases

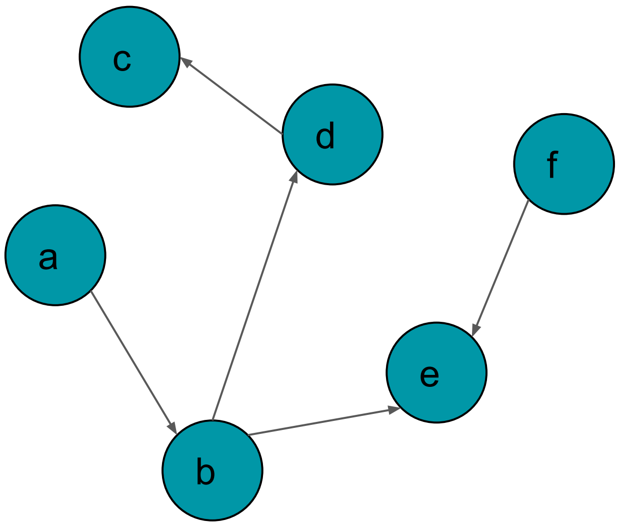
Predominant model for data storage and processing

Relational Databases

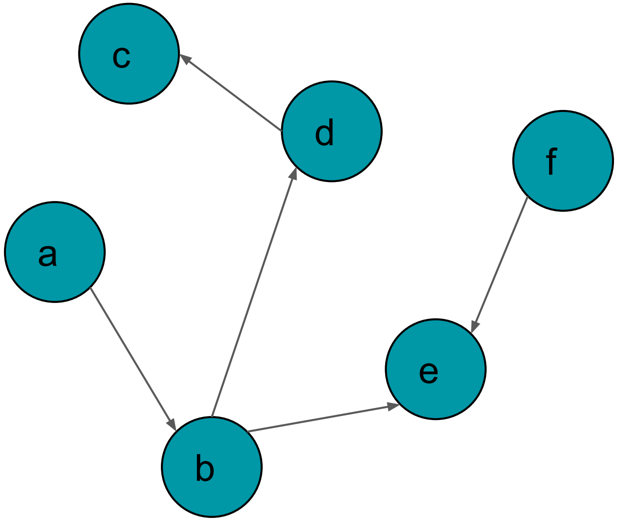
Predominant model for data storage and processing

Declarative language: focus on what rather than how

Relational Databases



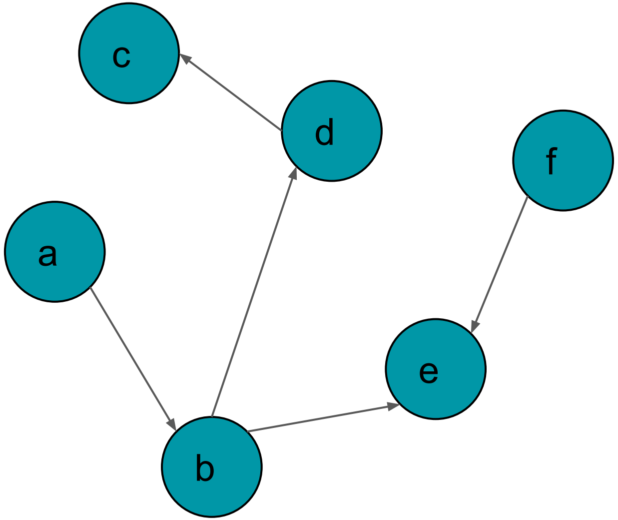
Relational Databases



INSERT INTO edges (...)

edges	
id_from	**id_to**
a	b
b	d
b	e
d	c
f	e

Relational Databases

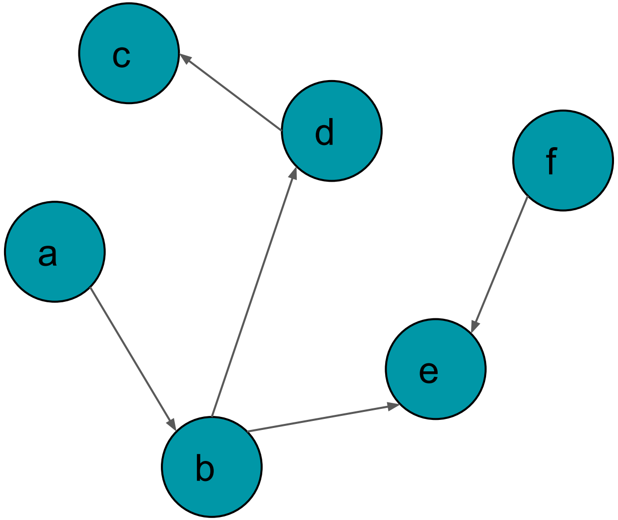


INSERT INTO edges (...)

edges	
id_from	**id_to**
a	b
b	d
b	e
d	c
f	e

Q: List vertices that vertex ‘b’ have an outgoing edge to.

Relational Databases



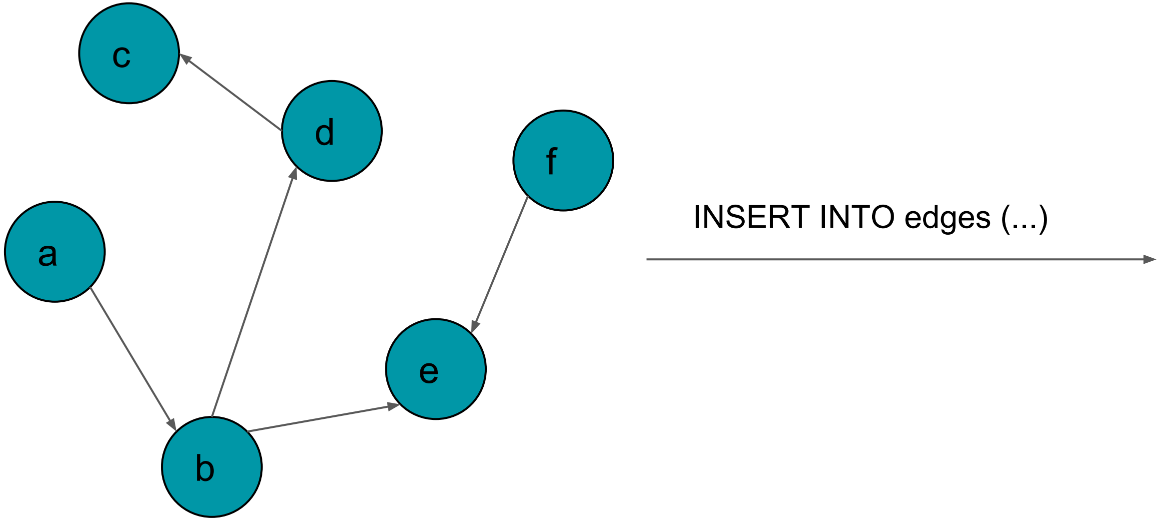
INSERT INTO edges (...)

edges	
id_from	**id_to**
a	b
b	d
b	e
d	c
f	e

Q: List vertices that vertex ‘b’ have an outgoing edge to.

A: SELECT id_to from edges WHERE id_from = ‘b’

Relational Databases



edges	
id_from	**id_to**
a	b
b	d
b	e
d	c
f	e

Q: List all vertex pairs (x,y), such that y is reachable from x.

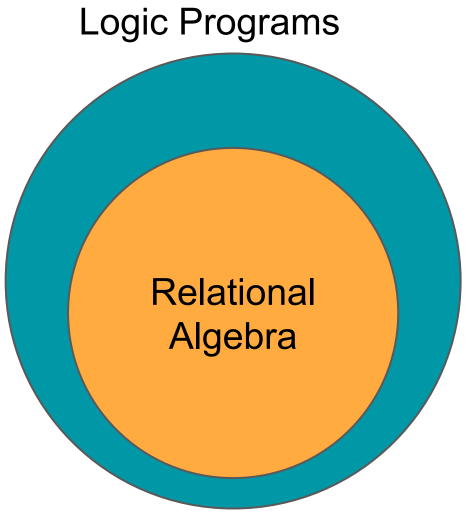
A: ?

Deductive Databases

Deductive Databases

Support a superset of relational algebra.

- Supports all queries from relational algebra.
- Supports recursions.



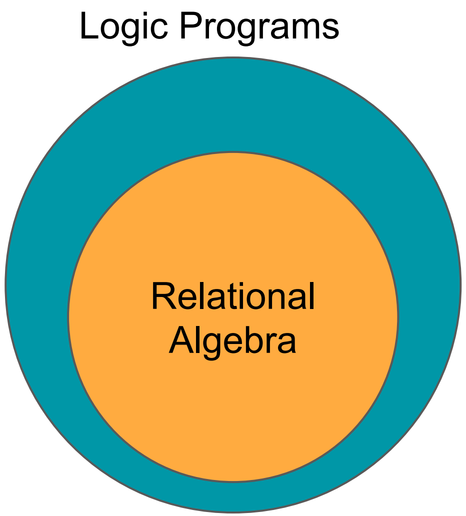
Deductive Databases

Support a superset of relational algebra.

- Supports all queries from relational algebra.
- Supports recursions.

Datalog: subset of Prolog, a logic programming language

- Database centric requirements
- Emphasis on completeness and termination
- Queries on data stored on secondary storage



Deductive Databases

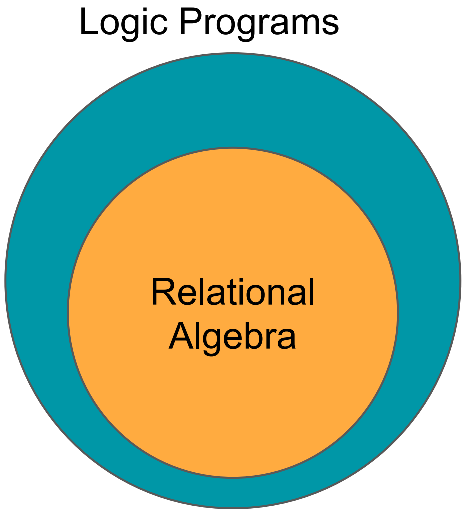
Support a superset of relational algebra.

- Supports all queries from relational algebra.
- Supports recursions.

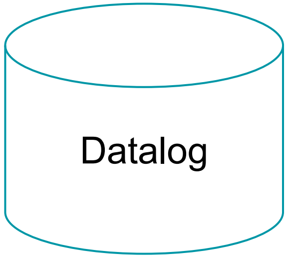
Datalog: subset of Prolog, a logic programming language

- Database centric requirements
- Emphasis on completeness and termination
- Queries on data stored on secondary storage

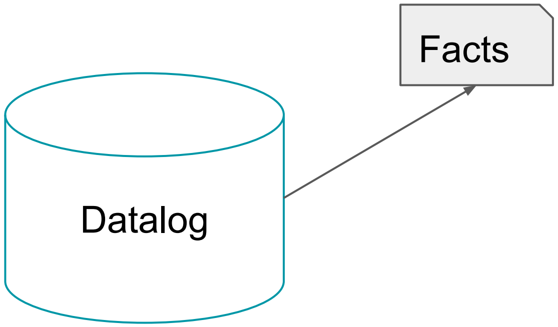
A database of facts.
A set of rules for deriving new facts from existing facts.



Datalog: Terminology

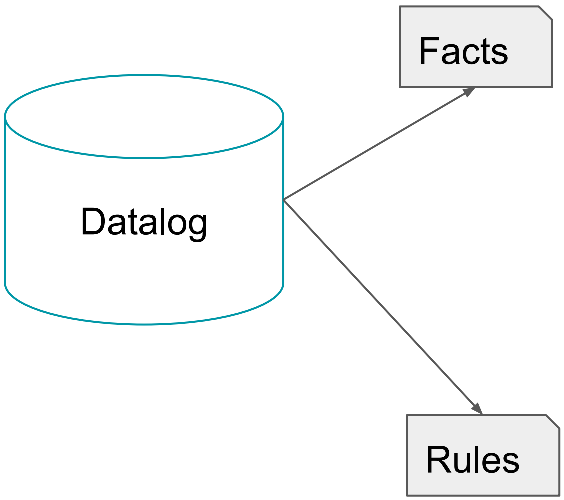


Datalog: Terminology



edge(a,b).

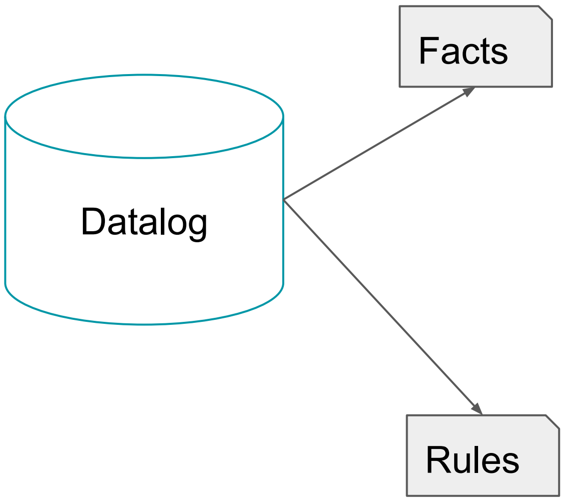
Datalog: Terminology




```
edge(a,b).
```

```
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).
```

Datalog: Terminology



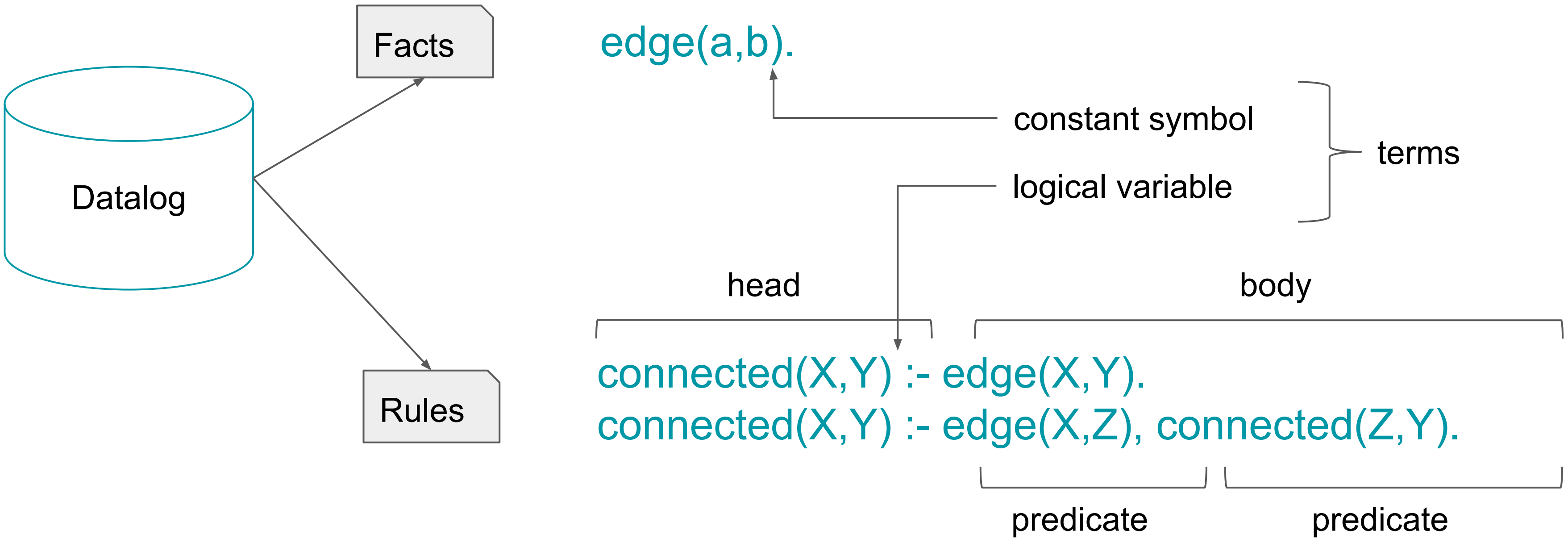
edge(a,b).

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

Implication/Clause: A_0 :- A_1, A_2, ..., A_k where A_0 is **true** if A_1 and A_2 … and A_k are **true.**

k = 0: fact; k > 0: rule

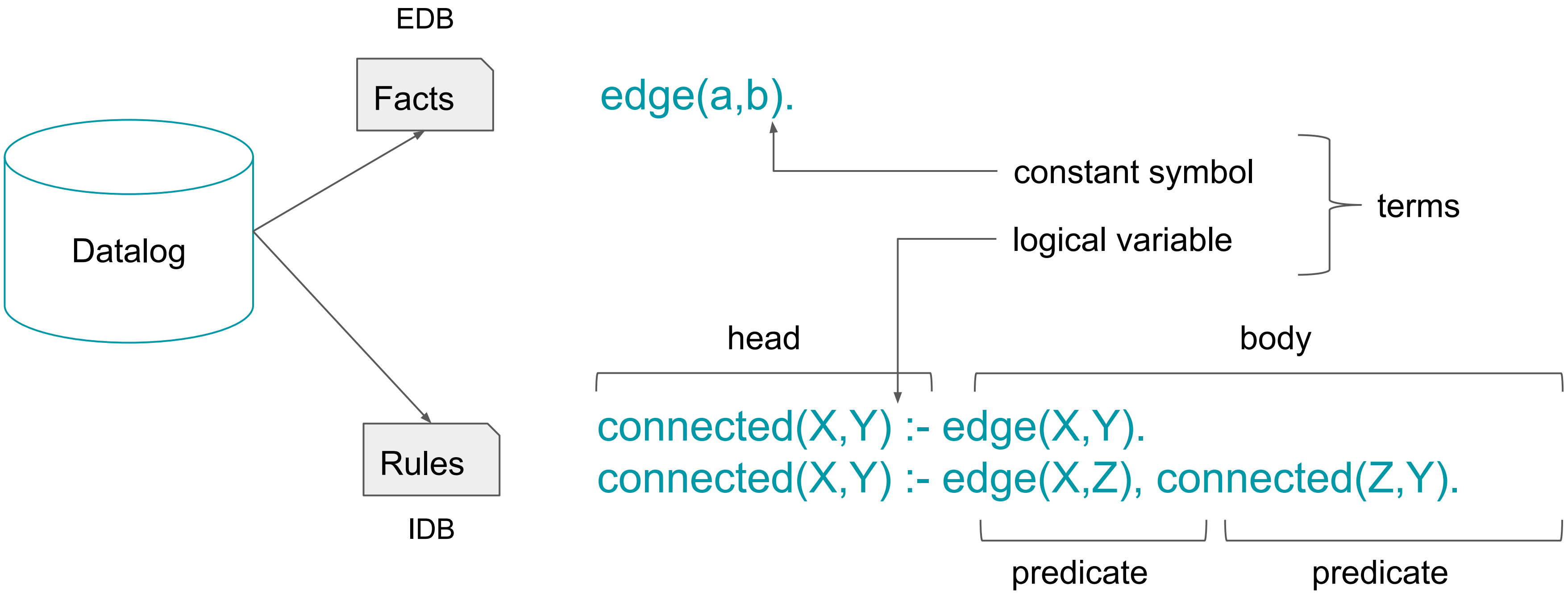
Datalog: Terminology



<u>Implication/Clause:</u> A_0 :- A_1, A_2, ..., A_k where A_0 is **true** if A_1 and A_2 … and A_k are **true.**

k = 0: fact; k > 0: rule

Datalog: Terminology



<u>Implication/Clause:</u> A_0 :- A_1, A_2, ..., A_k where A_0 is **true** if A_1 and A_2 … and A_k are **true.**

k = 0: fact; k > 0: rule

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Datalog: Examples

Selection

Q: List all users with **age** > 23.

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Selection

Q: List all users with **age** > 23.

Relational Algebra: $\sigma_{\textbf{age} > 23}(\text{users})$

SQL: SELECT * FROM users WHERE **age** > 23;

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Selection

Q: List all users with **age** > 23.

Relational Algebra: $\sigma_{\textbf{age} > 23}(\text{users})$

SQL: SELECT * FROM users WHERE **age** > 23;

Datalog: S(Uid, Name, Age) :- users(Uid, Name, Age), **Age** > 23.

Datalog: Examples

Projection

Q: List **name** of users with age > 23.

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Projection

Q: List **name** of users with age > 23.

Relational Algebra: $\pi_{\textbf{name}}(\sigma_{age > 23}(users))$

SQL: SELECT **name** FROM users WHERE age > 23;

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Projection

Q: List **name** of users with age > 23.

Relational Algebra: $\pi_{\textbf{name}}(\sigma_{age > 23}(users))$

SQL: SELECT **name** FROM users WHERE age > 23;

Datalog: P(**Name**) :- users(Uid, Name, Age), Age > 23.

Datalog: Examples

Join

Q: List **name, amount** of users with age > 23.

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Join

Q: List **name, amount** of users with age > 23.

Relational Algebra: $\pi_{\textbf{name,amount}}(\sigma_{age > 23}(users \bowtie_{uid} accounts))$

SQL:

SELECT **name,amount** FROM users,accounts

WHERE users.uid = accounts.uid AND age > 23;

Datalog: Examples

users
uid
name
age

accounts
uid
account_type
amount

users(42, ‘Jane Doe’, 26).

accounts(42, ‘savings’, 5692.23)

Join

Q: List **name, amount** of users with age > 23.

Relational Algebra: $\pi_{\textbf{name,amount}}(\sigma_{age > 23}(\text{users} \bowtie_{uid} \text{accounts}))$

SQL:

SELECT **name,amount** FROM users,accounts

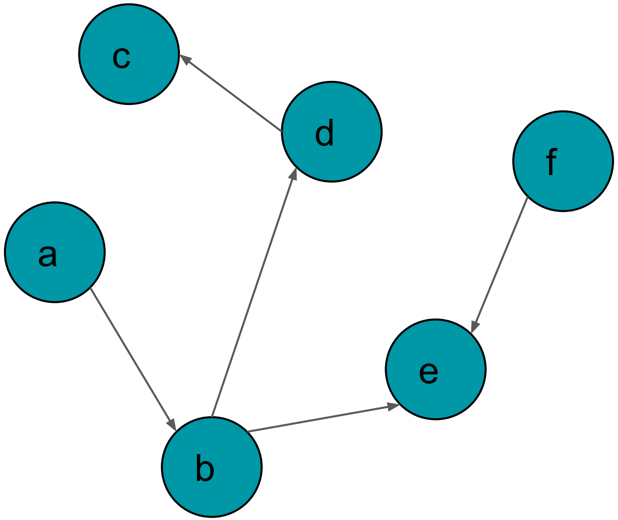
WHERE users.uid = accounts.uid AND age > 23;

Datalog:

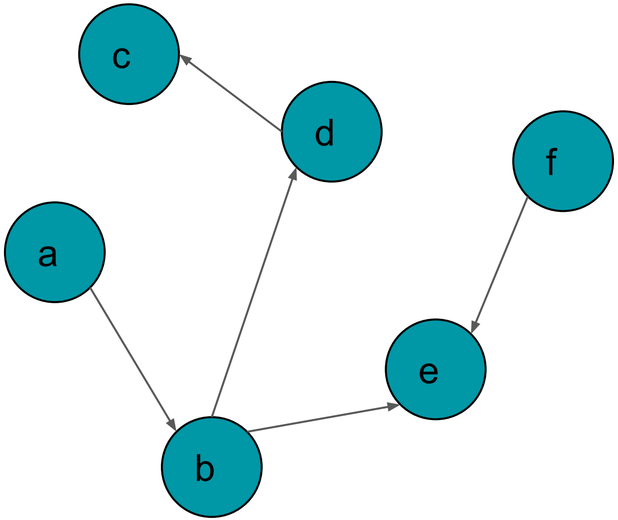
J(**Name,Amount**) :- users(Uid, Name, Age), accounts(Uid,

Account_type, Amount), Age > 23.

Datalog: Examples

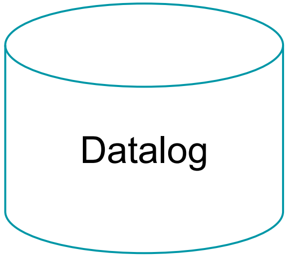


Datalog: Examples

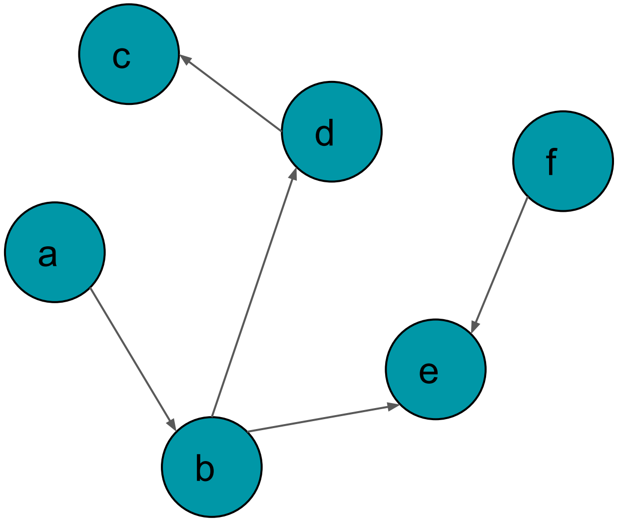



```
edge(a,b).
edge(b,d).
edge(b,e).
edge(d,c).
edge(f,e).
```

```
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).
```

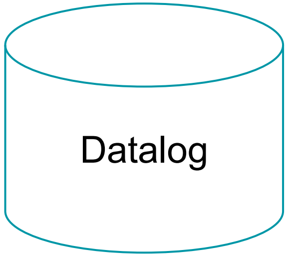


Datalog: Examples



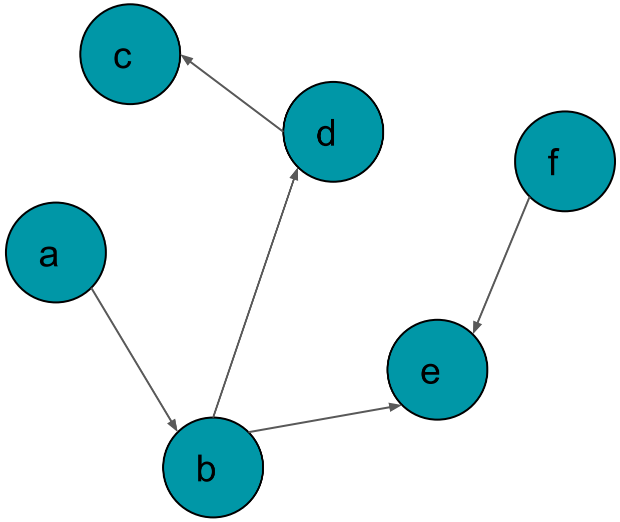
edge(a,b).
edge(b,d).
edge(b,e).
edge(d,c).
edge(f,e).

```
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).
```



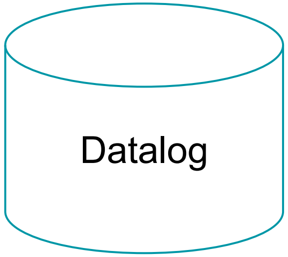
Q: List vertices that vertex ‘b’ have an outgoing edge to.

Datalog: Examples



edge(a,b).
edge(b,d).
edge(b,e).
edge(d,c).
edge(f,e).

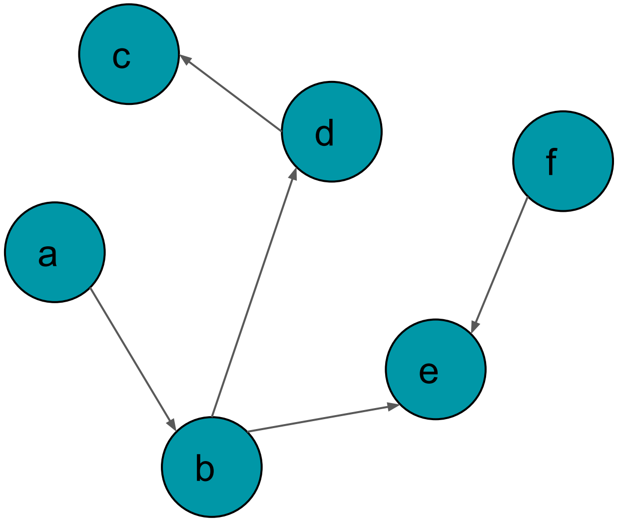
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).



Q: List vertices that vertex ‘b’ have an outgoing edge to.

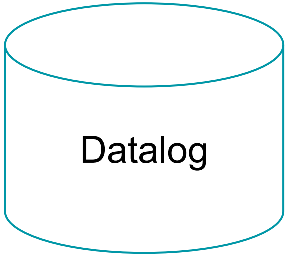
A: query(X) :- edge(b,X).

Datalog: Examples



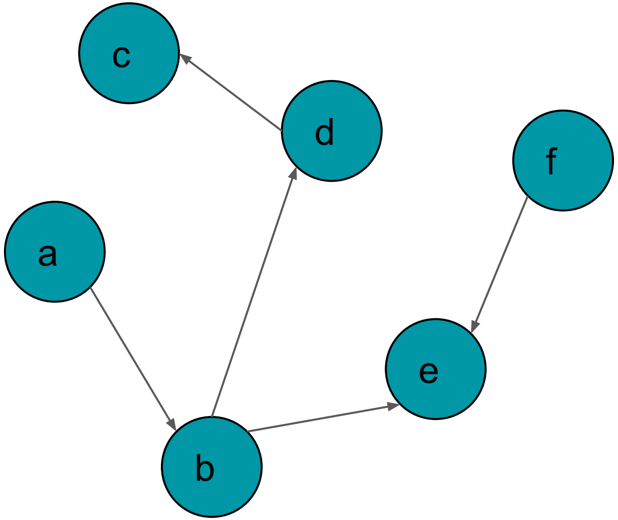
edge(a,b).
edge(b,d).
edge(b,e).
edge(d,c).
edge(f,e).

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).



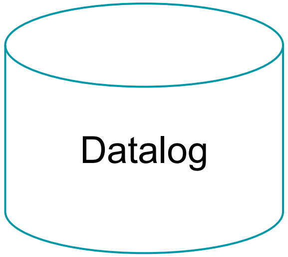
Q: List all vertex pairs (x,y), such that y is reachable from x.

Datalog: Examples



edge(a,b).
edge(b,d).
edge(b,e).
edge(d,c).
edge(f,e).

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).



Q: List all vertex pairs (x,y), such that y is reachable from x.

A: query(X,Y) :- connected(X,Y).

Query Evaluation: Naïve algorithm

Query Evaluation: Naïve algorithm

P_0 = InitialValue
Repeat
    $P_k = f(P_{k-1})$
Until *no-more-change*

http://www.cs.toronto.edu/~drosu/csc343-l7-handout6.pdf, http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/csep544_14wi_8/slide775.jpg

Query Evaluation: Naïve algorithm

1. Begin by assuming all IDB relations are empty.

P_0 = InitialValue
Repeat
    $P_k = f(P_{k-1})$
Until *no-more-change*

Query Evaluation: Naïve algorithm

1. Begin by assuming all IDB relations are empty.

2. Repeatedly evaluate the rules using the EDB and the previous IDB to get a new IDB.

P_0 = InitialValue
Repeat
    $P_k = f(P_{k-1})$
Until *no-more-change*

http://www.cs.toronto.edu/~drosu/csc343-l7-handout6.pdf, http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/csep544_14wi_8/slide775.jpg

Query Evaluation: Naïve algorithm

1. Begin by assuming all IDB relations are empty.

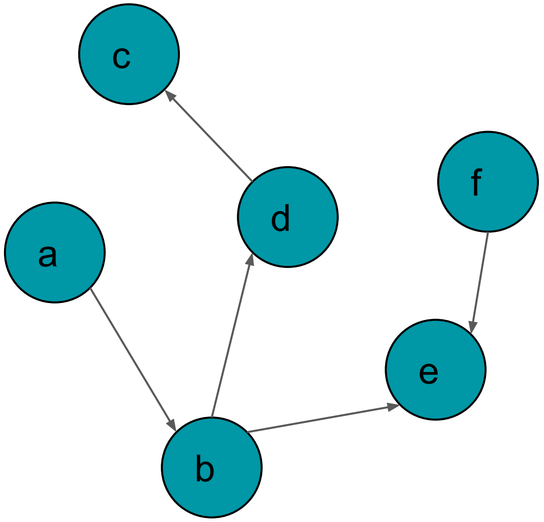
2. Repeatedly evaluate the rules using the EDB and the previous IDB to get a new IDB.

3. End when there is no change to the IDB.

P_0 = InitialValue
Repeat
    $P_k = f(P_{k-1})$
Until *no-more-change*

http://www.cs.toronto.edu/~drosu/csc343-l7-handout6.pdf, http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/csep544_14wi_8/slide775.jpg

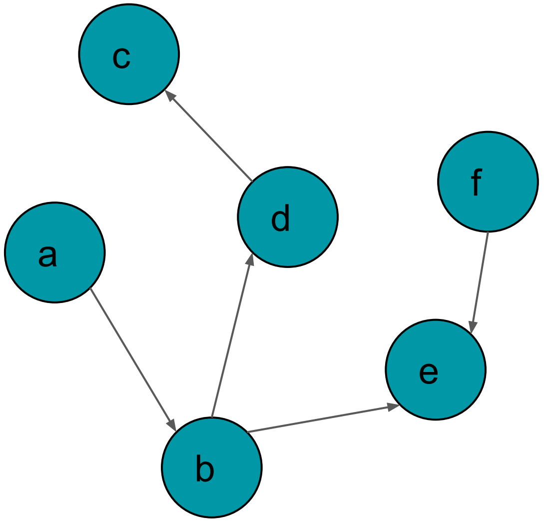
Query Evaluation: Naïve algorithm



connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

```
connected(X,Y).
```

Query Evaluation: Naïve algorithm



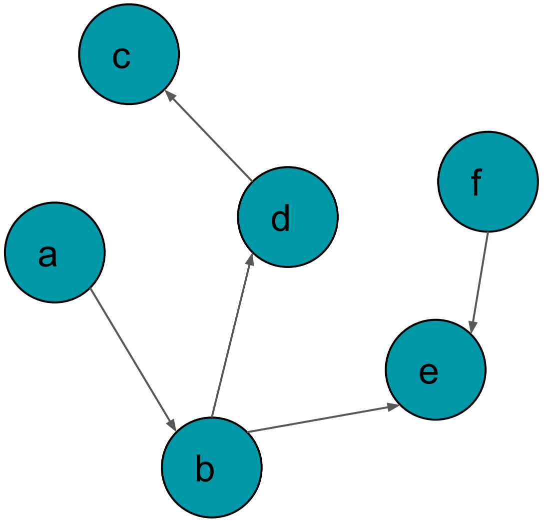
edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

```
connected(X,Y).
```

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	b
b	d
d	c
b	e
f	e

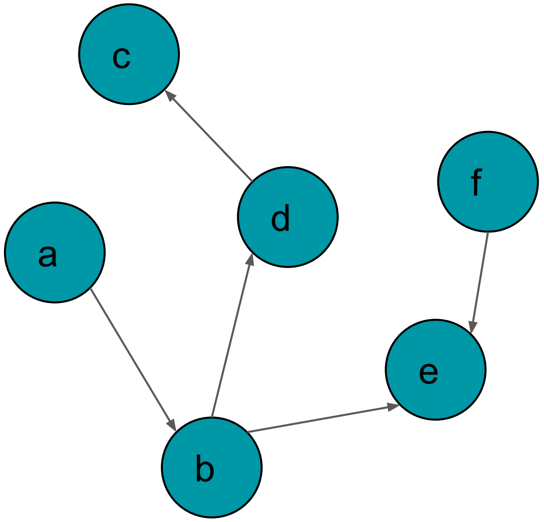
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

∅

I = 0

Query Evaluation: Naïve algorithm



edges	
a	**b**
b	**d**
d	**c**
b	**e**
f	**e**

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

∅

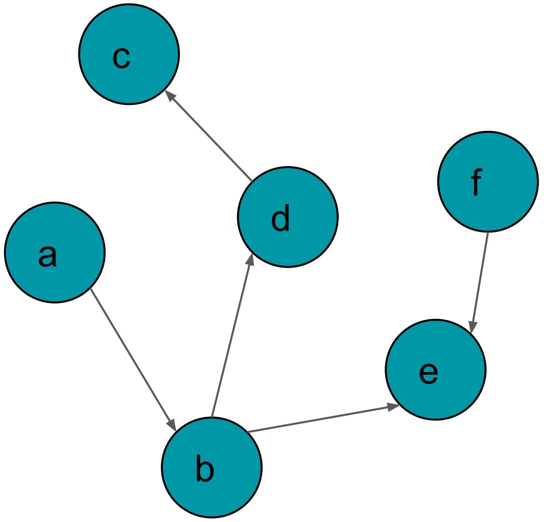
I = 0

a	b
b	d
d	c
b	e
f	e

I = 1

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

a	b
b	d
d	c
b	e
f	e

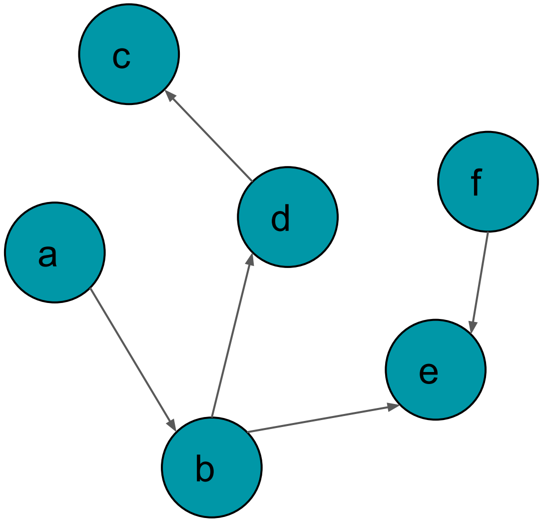
`connected(X,Y).`

∅

I = 0

I = 1

Query Evaluation: Naïve algorithm



edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

`connected(X,Y).`

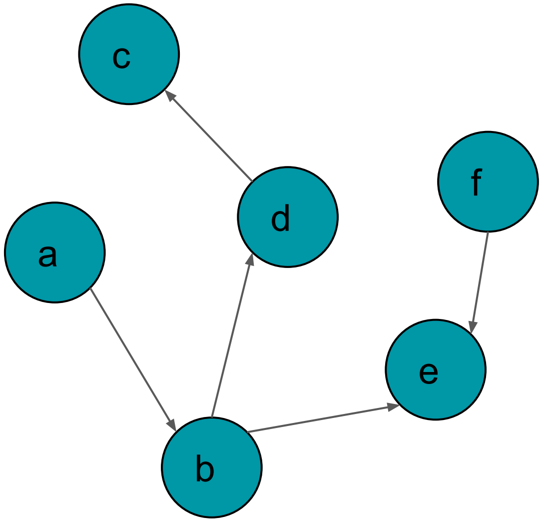
∅

a	b
b	d
d	c
b	e
f	e

I = 0 I = 1 I = 2

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	**b**
b	**d**
d	**c**
b	**e**
f	**e**

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

∅

I = 0

a	b
b	d
d	c
b	e
f	e

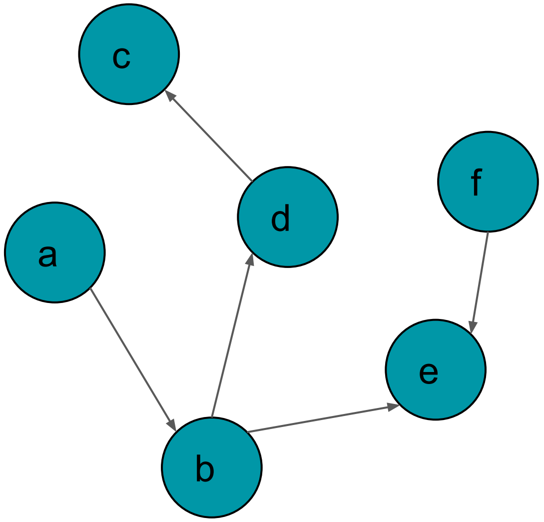
I = 1

a	b
b	d
d	c
b	e
f	e

I = 2

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	**b**
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

I = 0

∅

I = 1

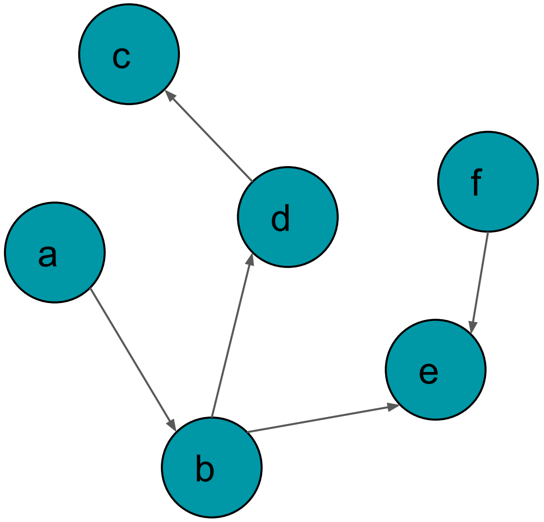
a	b
b	**d**
d	c
b	**e**
f	e

I = 2

a	d
a	e
a	b
b	d
d	c
b	e
f	e

Query Evaluation: Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).



edges	
a	b
b	**d**
d	c
b	e
f	e

```
connected(X,Y).
```

∅

I = 0

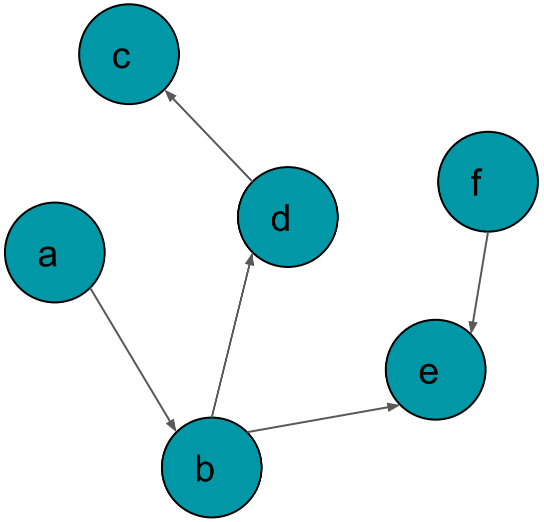
a	b
b	d
d	**c**
b	e
f	e

I = 1

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 2

Query Evaluation: Naïve algorithm



edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

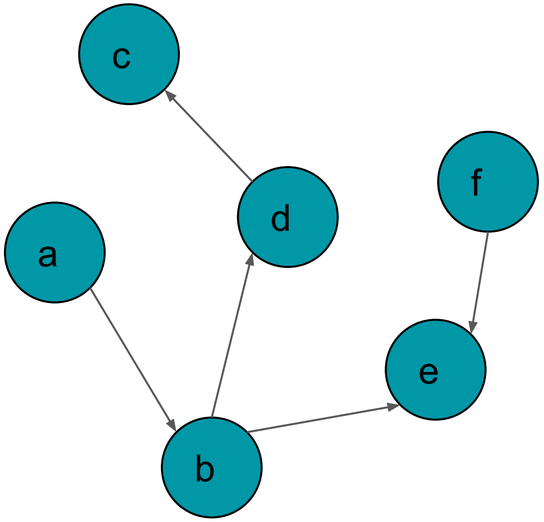
I = 2

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 3

Query Evaluation: Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).



edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

I = 2

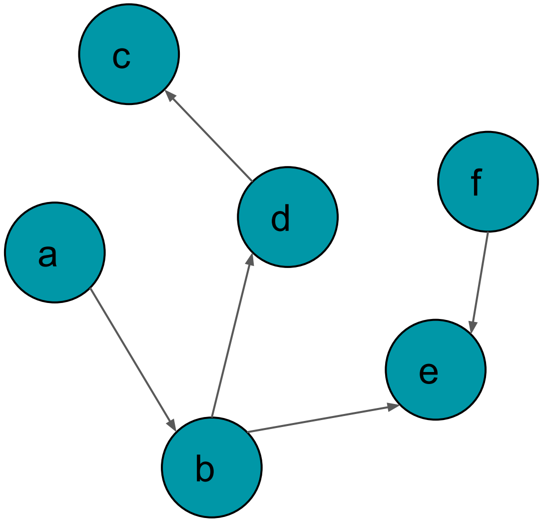
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 3

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	**b**
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

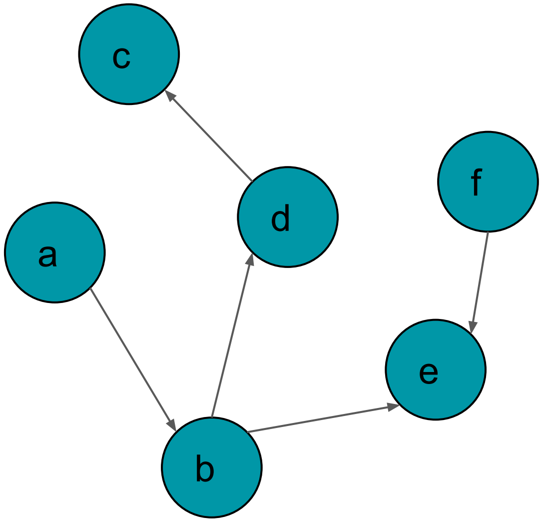
I = 2

b	**c**
a	d
a	e
a	b
b	**d**
d	c
b	**e**
f	e

I = 3

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

Query Evaluation: Naïve algorithm



edges	
a	b
b	**d**
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

I = 2

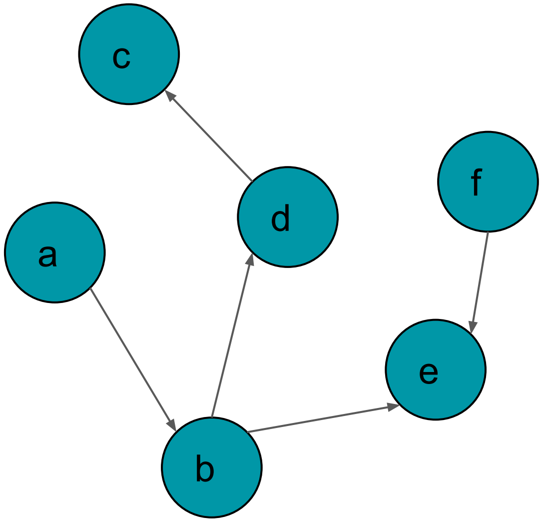
b	c
a	d
a	e
a	b
b	d
d	**c**
b	e
f	e

I = 3

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

I = 2

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 3

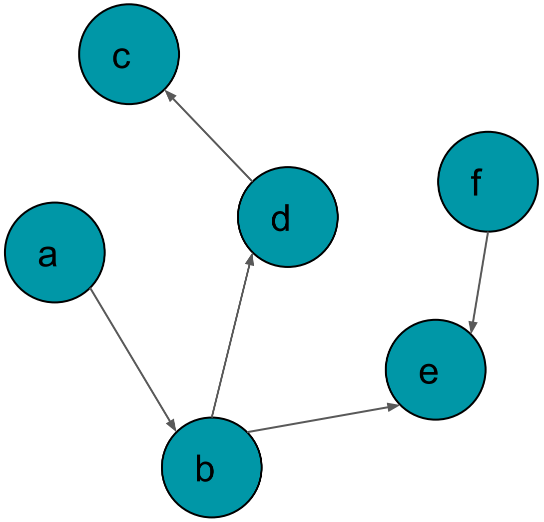
a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 4

```
connected(X,Y).
```

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



connected(X,Y).

edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

∅

I = 0

a	b
b	d
d	c
b	e
f	e

I = 1

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 2

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

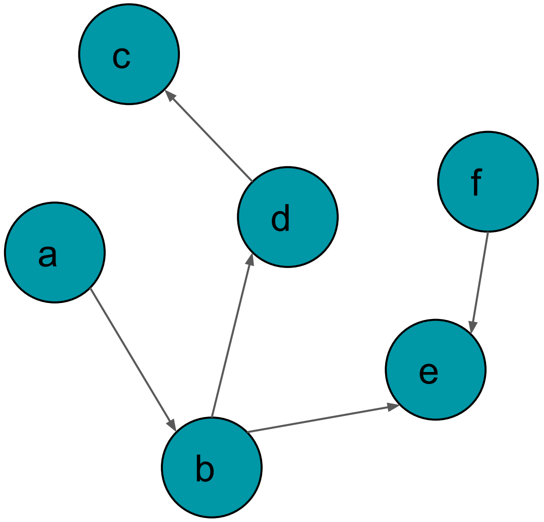
I = 3

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 4

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



connected(X,Y).

edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

I = 2

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 3

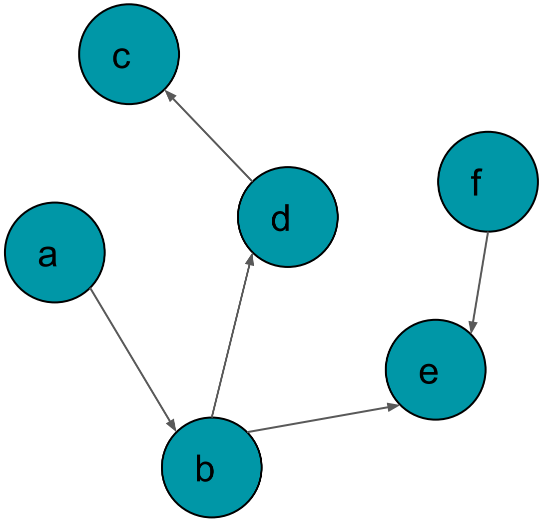
a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 4

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Naïve algorithm



edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

connected(X,Y).

I = 0

∅

I = 1

a	b
b	d
d	c
b	e
f	e

I = 2

b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 3

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

I = 4

a	c
b	c
a	d
a	e
a	b
b	d
d	c
b	e
f	e

http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/video.html?id=csep544_14wi_8

Query Evaluation: Semi-Naïve algorithm

* Avoid repeating computations already done in previous iterations.

* Focus on only the newly derived tuples (deltas) from previous iterations.

http://blogs.evergreen.edu/sosw/files/2014/04/Green-Vol5-DBS-017.pdf, http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/csep544_14wi_8/slide775.jpg

Query Evaluation: Semi-Naïve algorithm

* Avoid repeating computations already done in previous iterations.

* Focus on only the newly derived tuples (deltas) from previous iterations.

for each *IDB predicate* p

$$\textbf{do} \begin{cases} p^{[0]} := \emptyset \\ \delta(p)^{[0]} := \textit{tuples produced by rules using only EDB's} \end{cases}$$

$i := 1$

repeat

$p^{[i]} := p^{[i-1]} \cup \delta(p)^{[i-1]}$

evaluate $\Delta(p)^{[i]}$

$\delta(p)^{[i]} := \Delta(p)^{[i]} - p^{[i]}$

$i := i + 1$

until $\delta(p)^{[i]} = \emptyset$ *for each IDB predicate* p

http://blogs.evergreen.edu/sosw/files/2014/04/Green-Vol5-DBS-017.pdf, http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/csep544_14wi_8/slide775.jpg

Query Evaluation: Semi-Naïve algorithm

* Avoid repeating computations already done in previous iterations.

* Focus on only the newly derived tuples (deltas) from previous iterations.

for each *IDB predicate p*

$$\textbf{do} \begin{cases} p^{[0]} := \emptyset \\ \delta(p)^{[0]} := \textit{tuples produced by rules using only EDB's} \end{cases}$$

$i := 1$

repeat

$p^{[i]} := p^{[i-1]} \cup \delta(p)^{[i-1]}$

evaluate $\Delta(p)^{[i]}$

$\delta(p)^{[i]} := \Delta(p)^{[i]} - p^{[i]}$

$i := i + 1$

until $\delta(p)^{[i]} = \emptyset$ *for each IDB predicate p*

$$\Delta(p)^{[i]} \quad :- \quad \delta(p_1)^{[i-1]}, p_2^{[i-1]}, \ldots, p_n^{[i-1]}, q_1, \ldots, q_m .$$

$$\Delta(p)^{[i]} \quad :- \quad p_1^{[i]}, \delta(p_2)^{[i-1]}, p_3^{[i-1]} \ldots, p_n^{[i-1]}, q_1, \ldots, q_m .$$

$\ldots$

$$\Delta(p)^{[i]} \quad :- \quad p_1^{[i]}, \ldots, p_{n-1}^{[i]}, \delta(p_n)^{[i-1]}, q_1, \ldots, q_m .$$

http://blogs.evergreen.edu/sosw/files/2014/04/Green-Vol5-DBS-017.pdf, http://courses.cs.washington.edu/courses/csep544/14wi/video/archive/html5/csep544_14wi_8/slide775.jpg

Query Evaluation: Semi-Naïve algorithm

edges	
a	b
b	d
d	c
b	e
f	e

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

∅

$P_{I = 0}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I=0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I=0}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I=0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I = 0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I = 0}$

a	b
b	d
d	c
b	e
f	e

$P_{I = 1}$

a	d
a	e
b	c

$\Delta_{I = 1}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I=0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

a	d
a	e
b	c

$\Delta_{I=1}$

a	d
a	e
b	c

$\delta_{I=1}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I=0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

a	d
a	e
b	c

$\Delta_{I=1}$

a	d
a	e
b	c

$\delta_{I=1}$

a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

$P_{I=2}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I=0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

a	d
a	e
b	c

$\Delta_{I=1}$

a	d
a	e
b	c

$\delta_{I=1}$

a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

$P_{I=2}$

a	c

$\Delta_{I=2}$

Query Evaluation: Semi-Naïve algorithm

```
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).
```

edges	
a	b
b	d
d	c
b	e
f	e

∅

$P_{I=0}$

a	b
b	d
d	c
b	e
f	e

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

a	d
a	e
b	c

$\Delta_{I=1}$

a	d
a	e
b	c

$\delta_{I=1}$

a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

$P_{I=2}$

a	c

$\Delta_{I=2}$

a	c

$\delta_{I=2}$

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b	e
f	e

$P_{I=0}$: ∅

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

a	b
b	d
d	c
b	e
f	e

$\Delta_{I=1}$

a	d
a	e
b	c

$\delta_{I=1}$

a	d
a	e
b	c

$P_{I=2}$

a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

$\Delta_{I=2}$

a	c

$\delta_{I=2}$

a	c

a	c
a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

Query Evaluation: Semi-Naïve algorithm

connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(X,Z), connected(Z,Y).

edges	
a	b
b	d
d	c
b e	
f	e

$P_{I=0}$: ∅

$\delta_{I=0}$

a	b
b	d
d	c
b	e
f	e

$P_{I=1}$

a	b
b	d
d	c
b	e
f	e

$\Delta_{I=1}$

a	d
a	e
b	c

$\delta_{I=1}$

a	d
a	e
b	c

$P_{I=2}$

a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

$\Delta_{I=2}$

a	c

$\delta_{I=2}$

a	c

a	c
a	d
a	e
b	c
a	b
b	d
d	c
b	e
f	e

$\Delta_{I=3}$: ∅

$\delta_{I=3}$: ∅

Deductive Databases: Additional concepts

Deductive Databases: Additional concepts

Negation predicates

Deductive Databases: Additional concepts

Negation predicates

Safe rules

Deductive Databases: Additional concepts

Negation predicates

Safe rules

Query optimization
- Magic Sets
- Rule-Rewriting Techniques
- Iterative Fixpoint Evaluation

Deductive Databases: Additional concepts

Negation predicates

Safe rules

Query optimization
- Magic Sets
- Rule-Rewriting Techniques
- Iterative Fixpoint Evaluation

Aggregations

Conclusion

Conclusion

Deductive databases are more expressive than relational databases.

- Support for recursive queries

Conclusion

Deductive databases are more expressive than relational databases.

- Support for recursive queries

Datalog: query language adapted from Prolog

Conclusion

Deductive databases are more expressive than relational databases.

Support for recursive queries

Datalog: query language adapted from Prolog

Focus on query optimization

Conclusion

Deductive databases are more expressive than relational databases.

- Support for recursive queries

Datalog: query language adapted from Prolog

Focus on query optimization

Naive vs Semi Naive query execution algorithms

- Avoid repeated computations

Discussion

SQL has recursion techniques like CTE
- How does that compare to Datalog in terms of expressiveness?

Application domains best suited for Datalog?
- Program analysis (recursion)
- Declarative networking (NDlog)
- Security (SeNDlog)

Applicability to general processing frameworks?
- Hive
- Spark SQL