Updated 1/21/2020

T1: Data models and query languages
L4: Datalog

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp19)

https://northeastern-datalab.github.io/cs7240/sp20/

Version 1/17/2020

Where we are

Topic 1: Data models and query languages

- **Lecture 1 (Tue 1/7):** Course introduction, SQL refresher
  - Introduction, SQL
- **Lecture 2 (Fri 1/10):** Logic & relational calculus
  - SQL continued, Logic & relational calculus
- **Lecture 3 (Tue 1/14):** Relational Calculus, Relational algebra
  - Relational algebra
- **Lecture 4 (Fri 1/17):** Codd's theorem, Datalog
- **Lecture 5 (Tue 1/21):** Stable model semantics, Information theory & normal forms
- **Lecture 6 (Fri 1/24): (A1 due)** Alternative data models

Where We Are

- Relational query languages we have seen so far:
  - SQL
  - Relational Calculus
  - Relational Algebra
- They can express the same class of relational queries*
  - How powerful are they? What is missing?

* ignoring various extensions and semantic restrictions

Which are Relational Queries? Which are not? And Why?

- Given Friend(X,Y): Find all people X whose number of friends is a prime number ?

- Find all people who are friends with everyone who is not a friend of Bob

?

- Partition all people into three sets P1(X),P2(X),P3(X) s.t. any two friends are in different partitions

?

- Find all people who are direct or indirect friends with Alice (connected in arbitrary length)

?

Examples by Dan Suciu

Which are Relational Queries? Which are not? And Why?

- Given Friend(X,Y): Find all people X whose number of friends is a prime number

NO: needs higher math; not possible with RA

- Find all people who are friends with everyone who is not a friend of Bob

?

- Partition all people into three sets P1(X),P2(X),P3(X) s.t. any two friends are in different partitions

?

- Find all people who are direct or indirect friends with Alice (connected in arbitrary length)

?

Examples by Dan Suciu

Which are Relational Queries? Which are not? And Why?

- Given Friend(X,Y): Find all people X whose number of friends is a prime number

  NO: needs higher math; not possible with RA

- Find all people who are friends with everyone who is not a friend of Bob

  YES: example homework query

- Partition all people into three sets P1(X),P2(X),P3(X) s.t. any two friends are in different partitions

  ?

- Find all people who are direct or indirect friends with Alice (connected in arbitrary length)

  ?

Examples by Dan Suciu

Which are Relational Queries? Which are not? And Why?

- Given Friend(X,Y): Find all people X whose number of friends is a prime number

  NO: needs higher math; not possible with RA

- Find all people who are friends with everyone who is not a friend of Bob

  YES: example homework query

- Partition all people into three sets P1(X),P2(X),P3(X) s.t. any two friends are in different partitions

  NO: equivalent to 3-coloring; NP-complete

- Find all people who are direct or indirect friends with Alice (connected in arbitrary length)

  ?

Examples by Dan Suciu

Which are Relational Queries? Which are not? And Why?

- Given Friend(X,Y): Find all people X whose number of friends is a prime number

  NO: needs higher math; not possible with RA

- Find all people who are friends with everyone who is not a friend of Bob

  YES: example homework query

- Partition all people into three sets P1(X),P2(X),P3(X) s.t. any two friends are in different partitions

  NO: equivalent to 3-coloring; NP-complete

- Find all people who are direct or indirect friends with Alice (connected in arbitrary length)

  NO: recursive query; PTIME yet know expressible in RA

  Next: Datalog: extends RA with recursive queries

Datalog

- Database query language designed in the 80's
- Simple, concise, elegant
  - "Clean" restriction of Prolog with DB access
  - Expressive & declarative:
    - Set-of-rules semantics
    - Independence of execution order
    - Invariance under logical equivalence

```
Path(x,y) :- Edge(x,y)
Path(x,z) :- Edge(x,y), Path(y,z)
InCycle(x) :- Path(x,x)
```

- Few open source implementations, mostly academic implementations
- Today is a hot topic, beyond databases:
  - network protocols, static program analysis, DB+ML

Recursion with SQL server vs. Datalog

Proprietary SQL

Listing 4.7 Using Common Table Expressions for Recursive Operations

```
USE AdventureWorks;
WITH DirectReports (ManagerID, EmployeeID, EmployeeName, Title)
AS
(
-- Anchor member definition
  SELECT e.ManagerID, e.EmployeeID, c.FirstName + ' ' + c.LastName, e.Title
  FROM HumanResources.Employee AS e
  INNER JOIN Person.Contact as c
            ON e.ContactID = c.ContactID
  WHERE ManagerID IS NULL
  UNION ALL
-- Recursive member definition
  SELECT e.ManagerID, e.EmployeeID,c.FirstName + ' ' + c.LastName ,e.Title
  FROM HumanResources.Employee AS e
  INNER JOIN DirectReports AS d
    ON e.ManagerID = d.EmployeeID
  INNER JOIN Person.Contact as c
            ON e.ContactID = c.ContactID
)
-- Statement that executes the CTE
SELECT EmployeeID, EmployeeName, Title, ManagerID
FROM DirectReports
GO
```

Datalog

```
Manager(eid) :- Manages(_, eid)

DirectReports(eid, 0) :-
      Employee(eid), not Manager(eid)

DirectReports(eid, level+1) :-
      DirectReports(mid, level), Manages(mid, eid)
```

SQL Query vs. Datalog: which would you rather write?

Examples by Dan Suciu, Source of query on the left: Bieker, Lee. Mastering SQL server 2008

Outline: Datalog

- Datalog
  - Datalog rules
  - Recursion
  - Semantics
  - Datalog$^{\neg}$: Negation, stratification
  - Datalog$^{\pm}$
  - Stable model semantics (Answer set programming)
  - Datalog vs. RA
  - Naive and Semi-naive evaluation

Datalog: Facts and Rules

Schema

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

Facts: tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules: queries

```
Q1(y) :- Movie(x,y,z), z='1940'.
```

?

```
Q2(f,l) :- Actor(u,f,l), Casts(u,x),
           Movie(x,y,z), z<'1940'.
```

?

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                          Casts(z,x2), Movie(x2,y2,1940).
```

?

Datalog: Facts and Rules

Schema

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

Facts: tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules: queries

```
Q1(y) :- Movie(x,y,z), z='1940'.
```

Find movies from 1940

```
Q2(f,l) :- Actor(u,f,l), Casts(u,x),
           Movie(x,y,z), z<'1940'.
```

?

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                         Casts(z,x2), Movie(x2,y2,1940).
```

?

Examples by Dan Suciu

Datalog: Facts and Rules

Schema

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

Facts: tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules: queries

```
Q1(y) :- Movie(x,y,z), z='1940'.
```

Find movies from 1940

```
Q2(f,l) :- Actor(u,f,l), Casts(u,x),
           Movie(x,y,z), z<'1940'.
```

Find actors who played in a movie before 1940

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                          Casts(z,x2), Movie(x2,y2,1940).
```

?

Examples by Dan Suciu

Datalog: Facts and Rules

Schema

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

Facts: tuples in the database

```
Actor(344759,'Douglas', 'Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909, 'A Night in Armour', 1910).
Movie(29000, 'Arizona', 1940).
Movie(29445, 'Ave Maria', 1940).
```

Rules: queries

```
Q1(y) :- Movie(x,y,z), z='1940'.
```

Find movies from 1940

```
Q2(f,l) :- Actor(u,f,l), Casts(u,x),
           Movie(x,y,z), z<'1940'.
```

Find actors who played in a movie before 1940

```
Q3(f,l) :- Actor(z,f,l), Casts(z,x1), Movie(x1,y1,1910),
                          Casts(z,x2), Movie(x2,y2,1940).
```

Find actors who played in a movie from 1910 and from 1940

Extensional Database Predicates (EDB): Actor, Casts, Movie

Intensional Database Predicates (IDB): Q1, Q2, Q3

Examples by Dan Suciu

Syntax of rules

Q2(f,l) :- Actor(u,f,l), Casts(z,x), Movie(x,y,z), z<'1940'.

Alternative notations: Q(args) <- R1(args) AND R2(args) ....

Syntax of rules

Q2(f,l) :- Actor(u,f,l), Casts(z,x), Movie(x,y,z), z<'1940'.

head

(or *consequent*)
single *subgoal*

body

(or *antecedent*)
conjunction of *subgoals*

Alternative notations: Q(args) <- R1(args) AND R2(args) ....

Occasional convention: Variables begin with a capital, predicates begin with lower-case.

Syntax of rules

- evaluates to true when relation R_i contains the tuple described by $args_i$
- e.g. Actor(344759,'Douglas','Fowley') is true

$R_i(args_i)$: relational predicate with arguments ("atom")

arithmetic predicate

Q2(f,l) :- Actor(u,f,l), Casts(z,x), Movie(x,y,z), z<'1940'.

head

(or consequent)
single IDB subgoal

body

(or antecedent)
conjunction of subgoals

{f,l}: head variables
{u,x,y,z}: existential variables

Alternative notations: Q(args) <- R1(args) AND R2(args) ....

Occasional convention: Variables begin with a capital, predicates begin with lower-case.

Logical interpretation of a single rule

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

Q1(y) :- Movie(x,y,z), z<'1940'.

Meaning of a datalog rule is a logical statement:

?

Logical interpretation of a single rule

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

Q1(y) :- Movie(x,y,z), z<'1940'.

Meaning of a datalog rule is a logical statement:

For all x,y,z: if (x,y,z) ∈ Movies and z<'1940' then y is in Q1 (i.e. is part of the answer)

$$\forall x,y,z\ [(\text{Movie}(x,y,z) \land z<1940) \Rightarrow \text{Q1}(y)]$$

logically equivalent to

?

Logical interpretation of a single rule

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

Q1(y) :- Movie(x,y,z), z<'1940'.

Meaning of a datalog rule is a logical statement:

For all x,y,z: if (x,y,z) ∈ Movies and z<'1940' then y is in Q1 (i.e. is part of the answer)

$$\forall x,y,z\ [(\text{Movie}(x,y,z) \land z<1940) \Rightarrow \text{Q1}(y)]$$

logically equivalent to

$$\forall y\ [\exists x,z\ [\text{Movie}(x,y,z) \land z<1940] \Rightarrow \text{Q1}(y)\ ]$$

Thus, non-head variables are called "existential variables"

compare with RC

?

Logical interpretation of a single rule

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

Q1(y) :- Movie(x,y,z), z<'1940'.

Meaning of a datalog rule is a logical statement:

For all x,y,z: if (x,y,z) ∈ Movies and z<'1940' then y is in Q1 (i.e. is part of the answer)

$$\forall x,y,z\ [(\text{Movie}(x,y,z) \wedge z<1940) \Rightarrow \text{Q1}(y)]$$

logically equivalent to

$$\forall y\ [\exists x,z\ [\text{Movie}(x,y,z) \wedge z<1940] \Rightarrow \text{Q1}(y)]$$

Thus, non-head variables are called "existential variables"

We want the smallest set Q1 with this property (why?)

compare with RC

$$\{(y) \mid \exists x,z\ [\text{Movie}(x,y,z) \wedge z<1940]\ \}$$