Updated 2/8/2020

T1: Data models and query languages
L2: Logic, relational calculus

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

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

Version 1/10/2020

Queries and the connection to logic and algebra

- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

Logic in Computer Science and Databases

- Logic has had an immense impact on CS
- Computing has strongly driven one particular branch of logic: *finite model theory*
  - That is, First-order logic (FOL) restricted to finite models
  - Very strong connections to *complexity theory*
  - The basis of various branches in Artificial Intelligence
- It is a natural tool to capture and attack fundamental problems in database management
  - Relations as first-class citizens
  - Inference for assuring data integrity
  - *Inference for question answering (queries)*
- It has been used for developing and analyzing the relational model from the early days (Codd, 1972)

Why has Logics turned out to be so powerful?

- Basic Question: What on earth does an obscure, old intellectual discipline have to do with the youngest intellectual discipline?
- Cosma R. Shalizi, CMU:
  - "If, in 1901, a talented and sympathetic outsider had been called upon (say, by a granting-giving agency) to survey the sciences and name the branch that would be least fruitful in century ahead, his choice might well have settled upon mathematical logic, an exceedingly recondite field whose practitioners could all have fit into a small auditorium. It had no practical applications, and not even that much mathematics to show for itself: its crown was an exceedingly obscure definition of cardinal numbers."

Source: Moshe Vardi: Database Queries - Logic and Complexity

Back to The Future

- M. Davis (1988): Influences of Mathematical Logic on Computer Science:
  - “When I was a student, even the topologists regarded mathematical logicians as living in outer space. Today the connections between logic and computers are a matter of engineering practice at every level of computer organization.”

- Question: Why on earth?

Source: Moshe Vardi: Database Queries - Logic and Complexity

Birth of Computer Science: 1930s

- Church, Gödel, Kleene, Post, Turing: Mathematical proofs have to be “machine checkable” - computation lies at the heart of mathematics!
  - Fundamental Question: What is “machine checkable”?
- Fundamental Concepts:
  - algorithm: a procedure for solving a problem by carrying out a precisely determined sequence of simpler, unambiguous steps
  - distinction between hardware and software
  - a universal machine: a machine that can execute arbitrary programs
  - a programming language–notation to describe algorithms

Source: Moshe Vardi: Database Queries - Logic and Complexity

Leibniz's Dream

An Amazing Dream: a universal mathematical language, lingua characteristica universalis, in which all human knowledge can be expressed, and calculational rules, calculus ratiocinator, carried out by machines, to derive all logical relationships

- "If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, and say to each other: Calculemus–Let us calculate."

Source: Moshe Vardi: Database Queries - Logic and Complexity

Example: Aristotle' Syllogisms

- "All men are mortal"
- "For all x, if x is a man, then x is mortal"
- $(\forall x)(\text{Man}(x) \rightarrow \text{Mortal}(x))$

Source: Moshe Vardi: Database Queries - Logic and Complexity

Logic and Databases

Two main uses of logic in databases:

- Logic is used as a database query language to express questions asked against databases.

- Logic is used as a specification language to express integrity constraints in databases.

We will discuss mainly the first: query languages

Source: Phokion Kolaitis

Queries and the connection to logic and algebra

- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

First-Order Logic

- A formalism for specifying properties of mathematical structures, such as graphs, partial orders, groups, rings, fields, . . .
- Mathematical Structure:
  - $A = (D, R_1, \ldots, R_k, f_1, \ldots, f_l)$,
  - D is a non-empty set – universe, or domain
  - R_i is an m-ary relation on D, for some m (that is, $R_i \subseteq D^m$)
  - f_j is an n-ary function on D, for some n (that is, $f_i: D^n \rightarrow n$)

Source: Moshe Vardi: Database Queries - Logic and Complexity

First-Order Logic on Graphs

Syntax:

- First-order variables: x, y, z, . . . (range over nodes)
- Atomic formulas: E(x, y), x = y
- Formulas: Atomic Formulas + Boolean Connectives (∨, ∧, ¬) + First-Order Quantifiers (∃x, ∀x)

Source: Moshe Vardi: Database Queries - Logic and Complexity

Examples

- “node x has at least two distinct neighbors”

- “each node has at least two distinct neighbors”

Source: Moshe Vardi: Database Queries - Logic and Complexity

Examples

- “node x has at least two distinct neighbors”
  - $(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$
  - Concept: x is free in the above formula, which expresses a property of nodes.

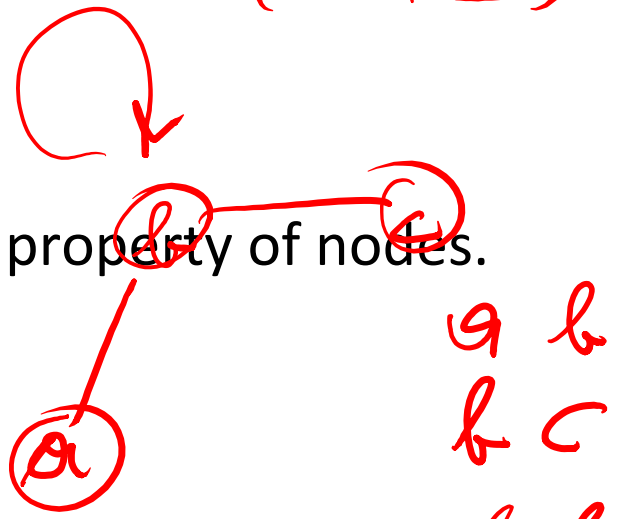
- “each node has at least two distinct neighbors”

Source: Moshe Vardi: Database Queries - Logic and Complexity

Examples

- "node x has at least two distinct neighbors"
  - $(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$
  - Concept: x is free in the above formula, which expresses a property of nodes.

- "each node has at least two distinct neighbors"
  - $(\forall x)(\exists y)(\exists z)(\neg(y = z) \wedge E(x, y) \wedge E(x, z))$
  - Concept: The above is a sentence, that is, a formula with no free variables; it expresses a property of graphs.



We will sometimes use $\exists x,y,z$ as short form for $\exists x \exists y \exists z$

Semantics of First-Order Logic

Semantics:

- First-order variables range over elements of the universes of structures
- To evaluate a formula φ, we need a graph G and a binding α that maps the free variables of φ to nodes of G

Notation: $G \models_{\alpha} \varphi(x_1, \ldots, x_k)$

Relational Databases

Codd's Two Fundamental Ideas:

- Tables are relations: a row in a table is just a tuple in a relation; order of rows/tuples does not matter!

- Formulas are queries: they specified the ***what*** rather then the ***how*** declarative programming!

Source: Moshe Vardi: Database Queries - Logic and Complexity

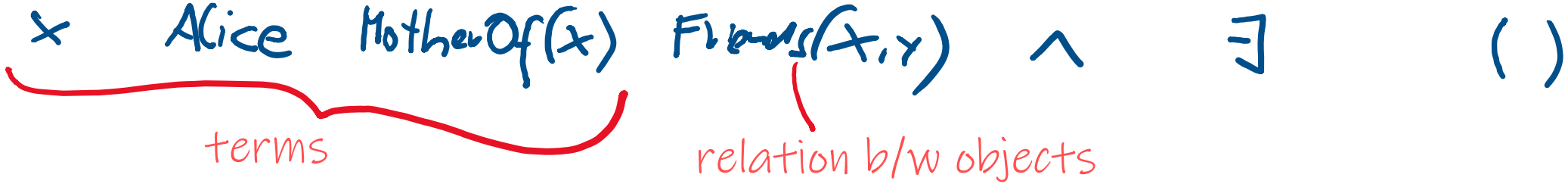
3 Components of FOL

1. Syntax (or language)
   - *What are the allowed syntactic expressions?*
   - DB world: schema, constraints, query language
2. Interpretation
   - *Mapping symbols to an actual world*
   - DB world: database
3. Semantics
   - *When is a statement "true" under some interpretation?*
   - DB world: meaning of constraint satisfaction and query results

Components of FOL: (1) Syntax

vocabulary

- Alphabet: symbols in use
  - Variables, constants, **function** symbols, **predicate** symbols, connectives, quantifiers, punctuation symbols



- Term: expression that stands for an *element or object*
  - Variable, constant
  - Inductively $f(t_1,\ldots,t_n)$ where t_i are terms, f a function symbol MotherOf(MotherOf(x))
- (Well-formed) formula: parameterized statement
  - *Atom* $p(t_1,\ldots,t_n)$ where p is a predicate symbol, t_i terms (atomic formula, together with predicates $t_1=t_2$)
  - Inductively, for formulas F, G, variable X:

$$F \wedge G \quad F \vee G \quad \neg F \quad F \longrightarrow G \quad F \longleftrightarrow G \quad \forall X F \quad \exists X F$$

- A *first-order language* refers to the set of all formulas over an alphabet

Also see ch 3.7.1 of Fagin et al: Reasoning about knowledge, 2003.

Components of FOL: (2) Interpretation

- An *interpretation* INT for an alphabet consists of:
  - A non-empty set $\mathbf{Dom}$, called domain
    - {Alice, Bob, Charly}
  - An assignment of an element in $\mathbf{Dom}$ to each constant symbol
    - Alice
  - An assignment of a function $\mathbf{Dom}^n \longrightarrow \mathbf{Dom}$ to each n-ary function symbol
    - Alice = MotherOf(Bob)
  - An assignment of a function $\mathbf{Dom}^n \longrightarrow \{\text{true}, \text{false}\}$ (i.e., a relation) to each n-ary predicate symbol
    - Friends(Bob, Charly) = TRUE

Components of FOL: (3) Semantics

- A variable assignment to a formula in an interpretation INT assigns to each *free variable* X a value from **Dom**
  - A *free variable* is one used without quantification

Person(X) ∃Y Married(X,Y)

bound

- Truth value for formula F under interpretation INT and variable assignment V:
  - Atom $p(t_1,\ldots,t_n)$: $q(s_1,\ldots,s_n)$ where q is the interpretation of the predicate p and s_i the interpretation of t_i
  - $F \wedge G$ $F \vee G$ $\neg F$ $F \longrightarrow G$ $F \longleftrightarrow G$: according to truth table
  - $\exists X F$: true iff there exists $d \in$ **Dom** such that if V assigns d to X then the truth value of F is true; otherwise false
  - $\forall X F$: true iff for all $d \in$ **Dom**, if V assigns d to X then the truth value of F is true; otherwise false
- If a formula has no free vars (closed formula or sentence), we can simply refer to its truth value under INT

∀X: Person(X) ⟶ Mortal(X)

Operator precedence

Operator precedence is an ordering of logical operators designed to allow the dropping of parentheses in logical expressions. The following table gives a hierarchy of precedences for the operators of propositional logic. The ¬ operator has higher precedence than ∧; ∧ has higher precedence than ∨; and ∨ has higher precedence than ⇒ and ⇔.

¬
∧
∨
⇒ ⇔

In unparenthesized sentences, it is often the case that an expression is flanked by operators, one on either side. In interpreting such sentences, the question is whether the expression associates with the operator on its left or the one on its right. We can use precedence to make this determination. In particular, we agree that an operand in such a situation always associates with the operator of higher precedence. When an operand is surrounded by operators of equal precedence, the operand associates to the right. The following examples show how these rules work in various cases. The expressions on the right are the fully parenthesized versions of the expressions on the left.

$\neg p \wedge q$	$((\neg p) \wedge q)$
$p \wedge \neg q$	$(p \wedge (\neg q))$
$p \wedge q \vee r$	$((p \wedge q) \vee r)$
$p \vee q \wedge r$	$(p \vee (q \wedge r)$
$p \Rightarrow q \Rightarrow r$	$(p \Rightarrow (q \Rightarrow r))$
$p \Rightarrow q \Leftrightarrow r$	$(p \Rightarrow (q \Leftrightarrow r))$

Source: http://intrologic.stanford.edu/glossary/operator_precedence.html

Queries and the connection to logic and algebra

- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

Entire Story in One Slide

1. RC = FOL over DB
2. RC can express “bad queries” that depend not only on the DB, but also on the domain from which values are taken [domain dependence]
3. We cannot test whether an RC query is “good,” but we can use a ”good” subset of RC that captures all “good” queries [safety]
4. “Good” RC and RA can express the same queries! [equivalence]

Relational Calculus (RC)

- RC is, essentially, first-order logic (FO) over the schema relations
  - A query has the form "find all tuples $(x_1,\ldots,x_k)$ that satisfy an FO condition"
- RC is a *declarative* query language
  - Meaning: a query is not defined by a sequence of operations, but rather by a condition that the result should satisfy

Queries and the connection to logic and algebra

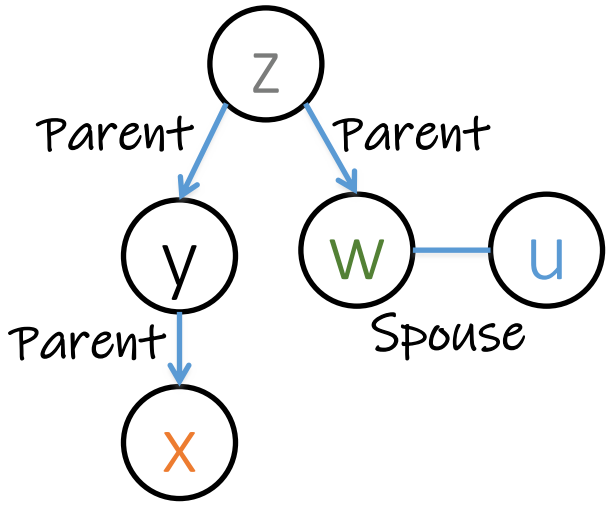
- Why logic?
  - A crash course on FOL
- Relational Calculus
  - Syntax and Semantics
  - Domain Independence and Safety
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RC and RA

RC Query

Person(id, gender, country)
Parent(parent, child)
Spouse(person1, person2)

assume symmetric relation
(a,b)∈Spouse ⇔ (b,a)∈Spouse

$$\{ (x,u) \mid \text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists y,z[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge$$
$$\exists w\, [\text{Parent}(z,w) \wedge y \neq w \wedge (u = w \vee \text{Spouse}(u,w))]\,]\,\}$$



Which relatives does this query find?

Example by Benny Kimelfeld

RC Symbols

- **Constant values**: a, b, c, female, Canada, ...
  - Values that may appear in table cells
- **Variables**: x, y, z, ...
  - Range over the values that may appear in table cells
- **Relation symbols**: R, S, T, Person, Parent, ...
  - Each with a specified arity
  - Will be fixed by the relational schema at hand
  - No attribute names, only attribute positions!
- Unlike general FOL, no function symbols!

Atomic RC Formulas

- Atomic formulas:
  - $R(t_1,\ldots,t_k)$
    - R is a k-ary relation
    - Each t_i is a variable or a constant
    - Semantically it states that $(t_1,\ldots,t_k)$ is a tuple in R

      Example: Person(x, 'female', 'Canada')

  - x op u
    - x is a variable, u is a variable/constant, op is one of >, <, =, ≠
    - Simply binary predicates, predefined interpretation

      Example: x=y, y≠w, z>5, z='female'

RC Formulas

- Formula:
  - Atomic formula
  - If φ and ψ are formulas then these are formulas:

$$\varphi \wedge \psi \quad \varphi \vee \psi \quad \varphi \rightarrow \psi \quad \varphi \rightarrow \psi \quad \neg\varphi \quad \exists x\, \varphi \quad \forall x\, \varphi$$

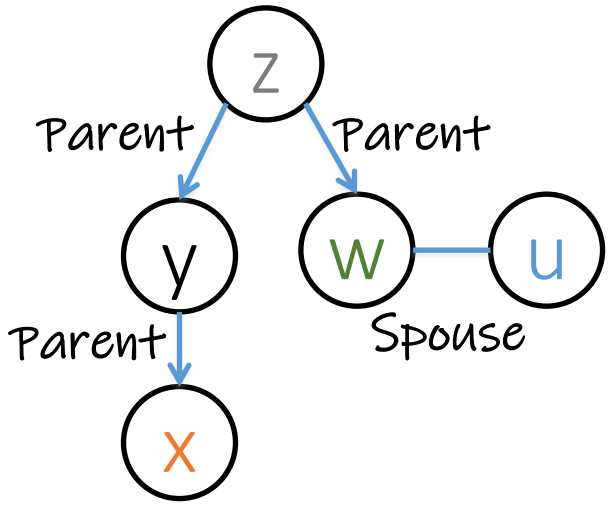
$$\begin{aligned}
&\text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge \\
&\exists y,z\big[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge \\
&\quad \exists w\, [\text{Parent}(z,w) \wedge y \neq w \wedge\ (u = w \vee \text{Spouse}(u,w))]\ \big]
\end{aligned}$$

Free Variables

- Intuitively: free variable are not bound to quantifiers
- Formally:
  - A free variable of an atomic formula is a variable that occurs in the atomic formula
  - A free variable of $\varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi$ is a free variable of either φ or ψ
  - A free variable of $\neg\varphi$ is a free variable of φ
  - A free variable of $\exists x\ \varphi$ and $\forall x\ \varphi$ is a free variable y of φ such that $y \neq x$
- We write $\varphi(x_1,\ldots,x_k)$ to state that $x_1,\ldots,x_k$ are the free variables of φ (in some order)

What Are the Free Variables?

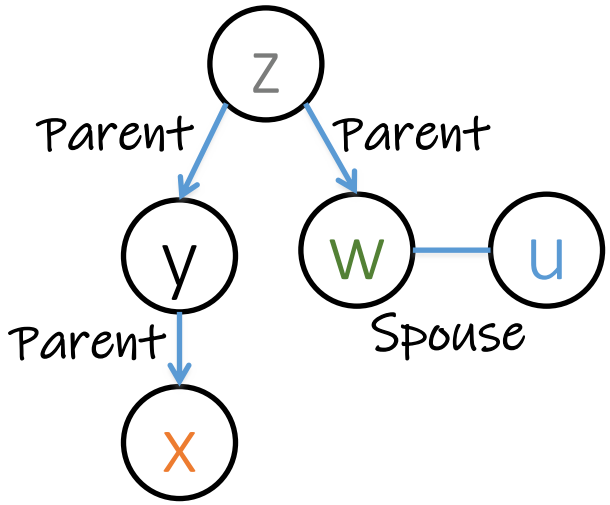
$$\text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists y,z\big[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge$$
$$\exists w\ [\text{Parent}(z,w) \wedge y \neq w \wedge\ (u = w \vee \text{Spouse}(u,w))]\ \big]$$



Example by Benny Kimelfeld

What Are the Free Variables?

$$\text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists y,z\big[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge$$
$$\exists w\,[\text{Parent}(z,w) \wedge y \neq w \wedge (u = w \vee \text{Spouse}(u,w))]\,\big]$$



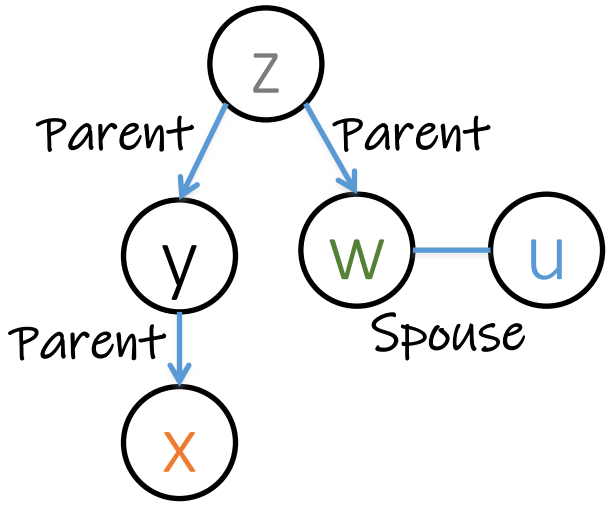
Notation:

$\varphi(x,u)$ / CanadianAunt(u,x)

Example by Benny Kimelfeld

RC query

$$\{ (x,u) \mid \text{Person}(u, \text{'female'}, \text{'Canada'}) \wedge$$
$$\exists y,z[\text{Parent}(z,y) \wedge \text{Parent}(y,x) \wedge$$
$$\exists w\ [\text{Parent}(z,w) \wedge y \neq w \wedge\ (u=w \vee \text{Spouse}(u,w))]\ ]\ \}$$



$$\{ (x_1,\ldots,x_k) \mid \varphi(x_1,\ldots,x_k) \}$$

$$\varphi(x,u) \;/\; \text{CanadianAunt}(u,x)$$

Example by Benny Kimelfeld

Relation Calculus Query

- An *RC query* is an expression of the form

$$\{ (x_1,\ldots,x_k) \mid \varphi(x_1,\ldots,x_k) \}$$

where $\varphi(x_1,\ldots,x_k)$ is an RC formula

some condition on the variables $COND(x_1,\ldots,x_k)$

- An RC query is *over* a relational schema $\mathbf{S}$ if all the relation symbols belong to $\mathbf{S}$ (with matching arities)

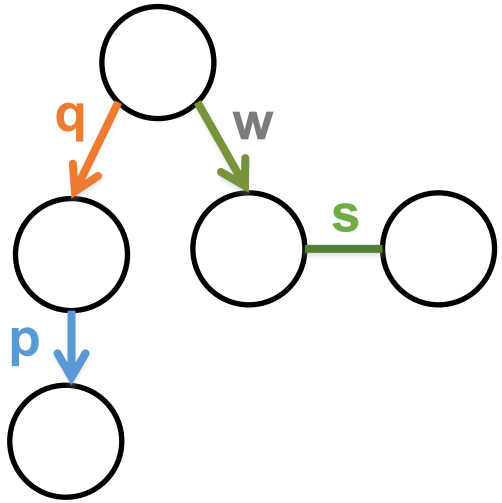
DRC vs. TRC

- There are two common variants of RC:
  - DRC: *Domain Relational Calculus* (what we're doing)
  - TRC: *Tuple Relational Calculus*
- DRC applies vanilla FO: terms interpreted as *attribute values*, relations have arity but no attribute names
- TRC is more "database friendly": terms interpreted as *tuples* with named attributes
- There are easy conversions between the two formalisms (nothing deep)

Our Example in TRC

Person(id, gender, country)
Parent(parent, child)
Spouse(person1, person2)

$$\{ t \mid \exists a\,[a \in \text{Person} \wedge a.\text{gender} = \text{'female'} \wedge a.\text{country} = \text{'Canada'}] \wedge$$
$$\exists p,q\,[p \in \text{Parent} \wedge p.\text{child} = t.\text{nephew} \wedge q \in \text{Parent} \wedge q.\text{child} = p.\text{parent}] \wedge$$
$$\exists w\,[w \in \text{Parent} \wedge w.\text{parent} = q.\text{parent} \wedge w.\text{child} \neq q.\text{child} \wedge$$
$$((t.\text{aunt} = w.\text{child} \wedge t.\text{aunt} = a.\text{id}) \vee \exists s\,[s \in \text{Spouse} \wedge$$
$$s.\text{person1} = w.\text{child} \wedge s.\text{person2} = t.\text{aunt} \wedge t.\text{aunt} = a[\text{id}]\ ]\ )\ ]\ ]\ ]\}$$

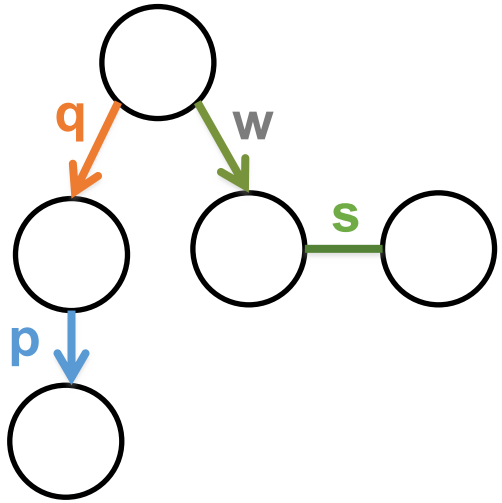


Notice we prefer and follow here the notation of [Ramakrishnan, Gehrke' 03] and [Elmasri, Navathe'15] of using **a.gender = 'female'** instead of the alternative notation **a[gender]='female'** used by [Silberschatz, Korth, Sudarshan 2010]

Our Example in TRC

Person(id, gender, country)
Parent(parent, child)
Spouse(person1, person2)

$$\{ t \mid \exists a \in \text{Person}\ [a.\text{gender} = \text{'female'} \wedge a.\text{country} = \text{'Canada'}] \wedge$$
$$\exists p,q \in \text{Parent}\ [p.\text{child} = t.\text{nephew} \wedge q.\text{child} = p.\text{parent}] \wedge$$
$$\exists w \in \text{Parent}\ [w.\text{parent} = q.\text{parent} \wedge w.\text{child} \neq q.\text{child} \wedge$$
$$((t.\text{aunt} = w.\text{child} \wedge t.\text{aunt} = a.\text{id}) \vee \exists s\ [s \in \text{Spouse} \wedge$$
$$s.\text{person1} = w.\text{child} \wedge s.\text{person2} = t.\text{aunt} \wedge t.\text{aunt} = a[\text{id}]\ ]\ )\ ]\ ]\ ]\}$$



often used short forms:

$\forall x \in R[\varphi]$ same as $\forall x[x \in R \Rightarrow \varphi]$

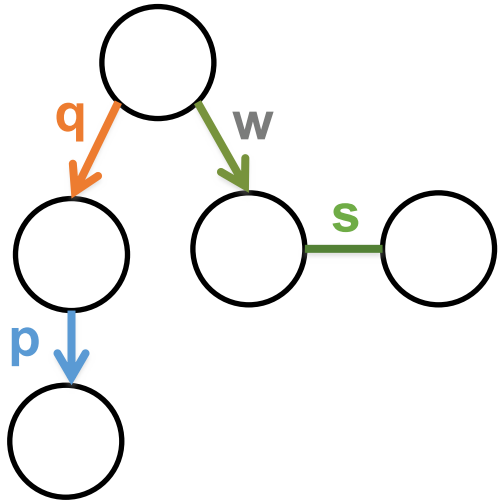
$\exists x \in R[\varphi]$ same as $\exists x[x \in R \wedge \varphi]$

Notice we prefer and follow here the notation of [Ramakrishnan, Gehrke' 03] and [Elmasri, Navathe'15] of using **a.gender = 'female'** instead of the alternative notation **a[gender]='female'** used by [Silberschatz, Korth, Sudarshan 2010]

Our Example in TRC

Person(id, gender, country)
Parent(parent, child)
Spouse(person1, person2)

$$\{ t \mid \exists a \in \text{Person}\ [a.\text{gender} = \text{'female'} \wedge a.\text{country} = \text{'Canada'}] \wedge$$
$$\exists p,q \in \text{Parent}\ [p.\text{child} = t.\text{nephew} \wedge q.\text{child} = p.\text{parent}] \wedge$$
$$\exists w \in \text{Parent}\ [w.\text{parent} = q.\text{parent} \wedge w.\text{child} \neq q.\text{child} \wedge$$
$$((t.\text{aunt} = w.\text{child} \wedge t.\text{aunt} \neq a.\text{id}) \vee \exists s\ [s \in \text{Spouse} \wedge$$
$$s.\text{person1} = w.\text{child} \wedge s.\text{person2} = t.\text{aunt} \wedge t.\text{aunt} = a[\text{id}]\ ])\ ]\ ]\ ]\}$$



tuple variables like in SQL instead of domain variables: {t | COND(t)}

Notice we prefer and follow here the notation of [Ramakrishnan, Gehrke' 03] and [Elmasri, Navathe'15] of using **a.gender = 'female'** instead of the alternative notation **a[gender]='female'** used by [Silberschatz, Korth, Sudarshan 2010]