These lecture notes include some material from Professors Bertossi, Kolaitis, Guagliardo, Vardi, Libkin, Barland, McMahan

Relational Calculus

Lecture Handout 4

Dr Evgenia Ternovska
Associate Professor

Simon Fraser University

Fall 2017

Relational Calculus

In addition to relational algebra, Codd introduced relational calculus.

Relational calculus is a declarative database query language based on first-order logic.

Relational calculus comes in two different flavours:

- Tuple relational calculus
- Domain relational calculus.

We will focus on domain relational calculus.

There is an easy translation between these two formalisms.

Codd's main technical result is that relational algebra and relational calculus have essentially the same expressive power.

Propositional Logic (aka Boolean Logic) Reminder

Propositional variables: $p, q, r, \ldots$
They take values 1 (True) and 0 (False).

Propositional connectives: $\wedge$, $\vee$, $\rightarrow$, $\neg$

Propositional formulas: expressions built from propositional variables and propositional connectives

Syntax: $\varphi := p, q, r, \ldots \mid (\psi \wedge \nu) \mid (\psi \vee \nu) \mid \neg\psi \mid (\psi \rightarrow \nu)$

Semantics: Truth-table semantics

Application: Propositional formulas express Boolean functions
$(p \vee q) \wedge (\neg p \vee \neg q)$ XOR-Gate
$(p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$ Majority Gate

First-order logic

Question: What is First-Order Logic?

Answer: Informally,

" First-Order Logic = Propositional Logic + ($\exists$ and $\forall$)",

where $\exists$ and $\forall$ range over possible values occurring in relations.

First-Order Logic: Syntax Reminder

$$
\begin{aligned}
\text{term } t &:= x && \text{(variable)} \\
&\mid c && \text{(constant)} \\
&\mid f(t_1, \ldots, t_n) && \text{(function application)}
\end{aligned}
$$

$$
\begin{aligned}
\text{formula } \varphi &:= P(t_1, \ldots, t_n) \\
&\mid t_1 \ \textbf{op} \ t_2 \quad \text{with } \textbf{op} \in \{=, \neq, >, <, \geqslant, \leqslant\} \\
&\mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \varphi_1 \to \varphi_2 \\
&\mid \exists x \ \varphi \mid \forall x \ \varphi \qquad \text{if } x \in \textbf{free}(\varphi)
\end{aligned}
$$

$\textbf{free}(\varphi) = \{$ variables that are not in the scope of any quantifier $\}$

Notation: sometimes we write $(x_1, \ldots, x_n)$ as $\bar{x}$.

Relational Calculus (First-Order Logic for Databases)

First-order variables: $x, y, z, \ldots, x_1, \ldots, x_k, \ldots$.
They **range over values that may occur in tables**.
Relation symbols:
$R, S, \ldots$ of specified arities (**names of relations**)
Atomic (Basic) Formulas:

- $R(x_1, \ldots, x_k)$, where R is a k-ary relation symbol (alternatively, $(x_1, \ldots, x_k) \in R$; the variables need not be distinct)
- $(x \ \textbf{op} \ y)$, where **op** is one of $=, \neq, <, >, \leq, \geq$
- $(x \ \textbf{op} \ c)$, where c is a constant and **op** is one of $=, \neq, <, >, \leq, \geq$.

Relational Calculus Formulas:
Every atomic formula is a relational calculus formula.
If φ_1 and φ_2 are relational calculus formulas, then so are:
$(\varphi_1 \wedge \varphi_2), (\varphi_1 \vee \varphi_2), (\neg\varphi), (\varphi_1 \to \varphi_2)$ (propositional connectives)
$(\exists x \varphi)$ (existential quantification)
$(\forall x \varphi)$ (universal quantification).

Free and bound variables

Examples: Assume E is a binary relation symbol

$$(\exists x)E(x,x)$$
$$(\forall x)(\forall y)(\exists z)(E(x,z) \wedge E(z,y))$$
$$(\exists z_1)(\exists z_2)(E(x,z_1) \wedge E(z_1,z_2) \wedge E(z_2,y))$$
$$(\exists y)(\exists z)(E(x,y) \wedge E(x,z) \wedge (y \neq z))$$

Free and bound variables:

In the first two formulas above, no variable is free.

In the third formula above, the free variables are x and y.

In the fourth formula above, the only free variable is x.

Intuitively, a variable is **free** in a formula if the variable must be **assigned a value** in order to tell if the formula is true or false.

Relational calculus as a Database Query Language (1)

A **relational calculus query** is an expression of the form

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

where the set of variables in $(x_1, \ldots, x_n)$ is free in φ, denoted **free**(φ)

Informally, when applied to a relational database D, a relational calculus query returns the k-ary relation that consists of all k-tuples $(a_1, \ldots, a_k)$ that make the formula "true" on D.

Example

The relational calculus expression $\{(x,y) : \exists z(E(x,z) \wedge E(z,y))\}$ returns the set P of all pairs of nodes (a,b) that are connected via a path of length 2.

Relational calculus as a Database Query Language (2)

Queries without free variables are called **Boolean queries**

Examples

- $Q = \{() \mid \forall x\ R(x, x)\}$
- $Q = \{() \mid \forall x\ \exists y\ R(x, y)\}$

Informally, their output is either **true** (denoted by $\{()\}$, a set containing the empty tuple) or **false** (denoted by $\varnothing$, the empty set of tuples)

We will define the formal semantics of relational queries shortly.

Examples

Customer : ID, Name, Age
Account : Number, Branch, CustID

Q_1: *Name of customers younger than 33 or older than 50*

$$\{\ (y) \mid \exists x, z\ \text{Customer}(x, y, z) \wedge (z < 33 \vee z > 50)\ \}$$

Q_2: *Name and age of customers having an account in London*

$$\{\ (y, z) \mid \exists x\ \text{Customer}(x, y, z) \wedge \exists w\ \text{Account}(w, \text{'London'}, x)\ \}$$

Q_3: *ID of customers who have an account in* **every** *branch*

$$\{\ (x) \mid \exists y, z\ (\text{Customer}(x, y, z)) \wedge \forall u, w, v\ (\text{Account}(u, w, v) \rightarrow \exists u'\ \text{Account}(u', w, x))\ \}$$

Natural Join in Relational Calculus

Example: Let R(A,B,C) and S(B,C,D) be two ternary relation schemas.
Recall that, in relational algebra, the natural join $R \bowtie S$ is given by

$$\pi_{R.A,R.B,R.C,S.D}(\sigma_{R.B=S.B \wedge R.C=S.C}(R \times S)).$$

Give a relational calculus expression for $R \bowtie S$:

$$\{(x_1, x_2, x_3, x_4) \mid R(x_1, x_2, x_3) \wedge S(x_2, x_3, x_4)\}$$

Note: The natural join is expressible by a quantifier-free formula of relational calculus. But we can also express it as

$$\{(x_1, x_2, x_3, x_6) \mid \exists x_4, x_5(x_4 = x_2 \wedge x_5 = x_3 \wedge R(x_1, x_2, x_3) \wedge S(x_4, x_5, x_6))\}$$

Quotient (Division) in Relational Calculus

Recall that the quotient (or division) $R \div S$ of two relations R and S is the relation of arity $r - s$ consisting of all tuples $(a_1, \ldots, a_{r-s})$ such that for every tuple $(b_1, \ldots, b_s)$ in S, we have that $(a_1, \ldots, a_{r-s}, b_1, \ldots, b_s)$ is in R.

Assume that R has arity 5 and S has arity 3. Express $R \div S$ in relational calculus.

$$\{(x_1, x_2) \mid \forall x_3 \forall x_4 \forall x_5(S(x_3, x_4, x_5) \rightarrow R(x_1, x_2, x_3, x_4, x_5))\}$$

Much simpler than the relational algebra expression for $R \div S$!

Semantics of First-order Logic

First-order **structure** $\mathcal{I}$ (often called Interpretation)

- non-empty domain of objects (universe) $\mathbf{U}$
- function $\cdot^{\mathcal{I}}$ that gives meaning (semantics) to constant/function/relation symbols (syntactic notions)

$$c^{\mathcal{I}} \in \mathbf{U}$$
$$f^{\mathcal{I}} \colon \mathbf{U}^n \to \mathbf{U}$$
$$R^{\mathcal{I}} \subseteq \mathbf{U}^n$$

A **database instance** D is a relational structure (no functions) (functions can be used in queries)

$D = (\mathbf{U}, P_1^D, P_2^D, \ldots, P_n^D)$ database instance over schema $\langle P_1, P_2, \ldots, P_n \rangle$

Example: $D = (\mathbf{U}, \text{Employee}^D)$ is a database instance over schema $\langle \text{Employee} \rangle$

Semantics of First-order Logic

Standard Name Assumption (SNA) for Databases:

Every constant is interpreted as itself: $c^{\mathcal{I}} = c$
That is, $\mathbf{U}$ just contains names of potential objects in the database.

Semantics defines whether a formula is true or false given a description of the world. In our case, the world will be a **database**. More precisely, semantics is itself a relation "$\models$" between formulae and worlds, denoted: $D \models \varphi$.

"D satisfies φ
"D models φ"
"φ holds true in D".

However, even if we are given a database, we can't tell whether $\text{Student}(x, \text{Math})$ holds or not, since we don't know what x is.

Semantics of First-order Logic: Assignment

Variable Assignment

- An assignment α is a function $\alpha : Var \to \mathbf{U}$, where Var is the set of variables in the formula and $\mathbf{U}$ is the domain of the database D.

Single point revision

Let α be an assignment, x be a variable, and d be an element in the domain $\mathbf{U}$. Then $\alpha[x \mapsto d]$ is an assignment defined as follows:

$$\alpha[x \mapsto d](z) = \begin{cases} d & \text{if } z = x, \\ \alpha(z) & \text{otherwise.} \end{cases}$$

Semantics of First-Order Logic

Semantics is a ternary relation between a formula φ, a database D, and an assignment $\alpha : Var \to \mathbf{U}$, where $\mathbf{U}$ is the domain of the database D.
We write this as "$D, \alpha \models \varphi$". Other notations for the same idea are "$D \models_\alpha \varphi$" and "$D \models \varphi[\alpha]$".

Semantics of atomic formulae

Let α be an assignment, let $\langle P_1, P_2, \ldots, P_n \rangle$ be a database schema (syntactic notion), where each P_i has arity k_i, and let $D = (\mathbf{U}, P_1^D, P_2^D, \ldots, P_n^D)$ be a database instance. Then

- $D, \alpha \models (t_1 = t_2)$ iff $\alpha(t_1) = \alpha(t_2)$.
- $D, \alpha \models P_i(t_1, t_2, \ldots, t_k)$ iff $(\alpha(t_1), \alpha(t_2), \ldots, \alpha(t_k)) \in P_i^B$, for any i in $\{1, \ldots, n\}$.

Semantics of Composite Formulae

- $D, \alpha \models \neg\varphi$ iff $D, \alpha \not\models \varphi$.
- $D, \alpha \models \varphi \wedge \psi$ iff $D, \alpha \models \varphi$ and $D, \alpha \models \psi$.
- $D, \alpha \models \varphi \vee \psi$ iff $D, \alpha \models \varphi$ or $D, \alpha \models \psi$.
- $D, \alpha \models \varphi \rightarrow \psi$ iff $D, \alpha \not\models \varphi$ or $D, \alpha \models \psi$.
- $D, \alpha \models \varphi \leftrightarrow \psi$ iff both $D, \alpha \models \varphi$ and $D, \alpha \models \psi$, or both $D, \alpha \not\models \varphi$ and $D, \alpha \not\models \psi$.
- $D, \alpha \models \exists x.\ \varphi$ iff there exists an $a \in \mathbf{U}$ such that $D, \alpha[x \mapsto a] \models \varphi$.
- $D, \alpha \models \forall x.\ \varphi$ iff for all $a \in \mathbf{U}$ such that $D, \alpha[x \mapsto a] \models \varphi$.

The last two cases are the only ones different from propositional logic. They are explained using modifications $\alpha[x \mapsto a]$ to the assignment α.

Examples

Let D be a database whose domain is $\{Abe, Joe, Zoe, Jill\} \cup \{Math, English\} \cup \mathbb{N}$, let α be an assignment which maps $x \mapsto Zoe$, and let the relation Employee^D be

EmployeeName	**Department**	**Manager**
Abe	Math	Charles
Joe	CS	Jill
Zoe	Math	Charles

For each of the following formulae, determine whether or not $D, \alpha \models \Theta$.

Examples

α maps $x \mapsto Zoe$, instance Employee^D:

EmployeeName	Department	Manager
Abe	Math	Charles
Joe	CS	Jill
Zoe	Math	Charles

- $\Theta = \neg$Employee (Zoe, Math, Jill) ("It's not the case that: Zoe is in the Math department and managed by Jill.")
- $\Theta = \exists y.$Employee(y, Math, Jill)
  ("Somebody is in the math department and managed by Jill")
- $\Theta =$ (Employee (x, Math, Jill) $\wedge$ Employee (z, Math, Jill))
  ("x and z both work in the math department and are both managed by Jill").

Semantics: Answers to queries

- Fix an underlying domain $\mathbf{U}$ under SNA
- Fix an underlying structure, an interpretation of relation and function symbols

  Recall that "interpretations" are just databases

Recall: an assignment α maps variables to objects in $\mathbf{U}$

The **answer** to a query $Q = \{\bar{x} \mid \varphi\}$ on a database D is

$$Q(D) = \{\ \alpha(\bar{x}) \mid \alpha\colon \textbf{free}(\varphi) \to \mathbf{U} \text{ such that } D, \alpha \models \varphi\ \}$$

where $D, \alpha \models \varphi$ means φ is true in D with variable assignment α

The **answer** to a **Boolean query** is either $\{()\}$ (true) or $\varnothing$ (false)

Relational Algebra vs. Relational Calculus

Fundamental theorem of database theory

Codds Theorem (informal statement):

Relational Algebra and Relational Calculus have essentially the same expressive power, i.e., they can express the same queries.

Note:

- This statement is **not** entirely accurate.
- In what follows, we will give a rigorous formulation of Codds Theorem and sketch its proof.

($\Rightarrow$) From algebra to calculus

($\Rightarrow$) Theorem 1:

For every relational expression E, there is an equivalent relational calculus expression $\{(x_1, \ldots, x_k) \mid \varphi(x_1, \ldots, x_k)\}$.

Proof (sketch): Translation, by induction on the construction of relational algebra expressions.

The translation is given in the next few slides.

($\Rightarrow$) From algebra to calculus

Translate each RA expression E into a first-order formula φ

Assumption: the attributes of a relation are **ordered**
(R over A, B, C means the 1st column is A, the 2nd is B, the 3rd is C)

Environment η (a function)
maps each attribute A in the schema to a variable x_A

($\Rightarrow$) From algebra to calculus

Base relation

R over $A_1, \ldots, A_n$ is translated to $R\big(\eta(A_1), \ldots, \eta(A_n)\big)$

Example
If R is a base relation over A, B

$$\eta = \{\ A \mapsto x_A,\ B \mapsto x_B,\ \ldots\ \}$$

then R is translated to $R(x_A, x_B)$

($\Rightarrow$) From algebra to calculus

Renaming $\rho_{A \to B}(E)$

1. Translate E to φ
2. If there is no mapping for B in η, add $\{B \mapsto x_B\}$
3. Replace every occurrence of $\eta(A)$ in φ by $\eta(B)$

Example

If R is a base relation over A, B
then $\rho_{A \to B}\big(\rho_{B \to C}(R)\big)$ is translated to $R(x_B, x_C)$

($\Rightarrow$) From algebra to calculus

Projection

$\pi_\alpha(E)$ is translated to $\exists X\ \varphi$

where

- φ is the translation of E
- $X = \textbf{free}(\varphi) - \eta(\alpha)$
  (attributes that are **not** projected become quantified)

Example

If R is a base relation over A, B
then $\pi_A(R)$ is translated into $\exists x_B\ R(x_A, x_B)$

($\Rightarrow$) From algebra to calculus

Selection

$$\sigma_\theta(E) \text{ is translated to } \varphi \wedge \eta(\theta)$$

where

- φ is the translation of E
- $\eta(\theta)$ is obtained from θ by replacing each attribute A by $\eta(A)$

Example

If R is a base relation over A, B
then $\sigma_{A=B}(R)$ is translated into $R(x_A, x_B) \wedge x_A = x_B$

($\Rightarrow$) From algebra to calculus

Cartesian Product, Union, Difference

Product	$E_1 \times E_2$	is translated to	$\varphi_1 \wedge \varphi_2$
Union	$E_1 \cup E_2$	is translated to	$\varphi_1 \vee \varphi_2$
Difference	$E_1 - E_2$	is translated to	$\varphi_1 \wedge \neg\varphi_2$

where

- φ_1 is the translation of E_1
- φ_2 is the translation of E_2

In fact, our translation gives us what is called a *safe* relational calculus (explained below).

Corollary: Relational Calculus is **relationally complete**.

From Relational Calculus to Relational Algebra

Fact: It is **not** true that for every relational calculus expression φ, there is an equivalent relational algebra expression E.

Examples:

- $\{(x) \mid \neg R(x)\}$
- $\{(x, y) \mid R(x) \vee S(y)\}$
- $\{(x, y) \mid x = y\}$
- $\{(x, y) \mid \exists z(\text{CHAIR}(x, z) \wedge y \neq z)\}$, where CHAIR(dpt,name)
- $\{(x) \mid \forall y \forall z(\ \text{ENROLLS}(x, y, z))\}$,
  where ENROLLS(s-name,course,term)

From Relational Calculus to Relational Algebra

A query is **safe** if it gives a finite answer on **all** databases

The queries above are not safe.

Bad news:
Whether a relational calculus query is safe is **undecidable**

Question:
How can we go from relational calculus to relational algebra?

Answer:
"Relativize" the semantics of relational calculus expressions by fixing a domain over which the variables range.

Active domain

Adom$(R) = \{$ all constants occuring in $R \}$

Example

$$\textbf{Adom}\left(\begin{array}{c|cc} R & A & B \\ \hline & a_1 & b_1 \\ & a_1 & b_2 \end{array}\right) \quad = \quad \{a_1, b_1, b_2\}$$

The active domain of a database D is

$$\textbf{Adom}(D) = \bigcup_{R \in D} \textbf{Adom}(R)$$

Active domain and safety

For a safe query Q, we have that $\textbf{Adom}\big(Q(D)\big) \subseteq \textbf{Adom}(D)$

Active domain semantics

Evaluate queries within **Adom**(D) $\implies$ **safe relational calculus**

$$Q(D) = \{\ \alpha(\bar{x}) \mid \alpha\colon \textbf{free}(\varphi) \to \textbf{Adom}(D) \text{ s.t. } D, \alpha \models \varphi\ \},$$

where $D, \alpha \models \varphi$ means φ is true in D with variable assignment α

For each $\alpha\colon \textbf{free}(\varphi) \to \textbf{Adom}(D)$ (there are finitely many such α)
$Q(D)$ outputs $\alpha(\bar{x})$ whenever $D, \alpha \models \varphi$

($\Leftarrow$) From calculus to algebra

($\Leftarrow$) Theorem 2:

For every **safe** relational calculus expression

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

there is an equivalent relational algebra expression E.

Proof (sketch): Translation, by induction on the construction of relational calculus expressions.

The translation is given in the next few slides.

($\Leftarrow$) From calculus to algebra

Translate each FOL formula φ into an RA expression E

Assumptions (without loss of generality)

- No universal quantifiers, implications, double negations
- No distinct pair of quantifiers binds the same variable
- No variable occurs both free and bound
- No variable is repeated within a predicate
- No constants in predicates
- No atoms of the form x **op** x or c_1 **op** c_2

Environment η (a function)

maps each variable x to an attribute A_x

($\Leftarrow$) From calculus to algebra

Let R be over attributes $A_1, \ldots, A_n$

Predicate symbols

$R(x_1, \ldots, x_n)$ is translated to $\rho_{A_1 \to \eta(x_1), \ldots, A_n \to \eta(x_n)}(R)$

Example

For R over attributes A, B, C,
$R(x, y, z)$ is translated into $\rho_{A \to A_x, B \to A_y, C \to A_z}(R)$

($\Leftarrow$) From calculus to algebra

Existential quantification

$\exists x\ \varphi$ is translated $to\, \pi_{\eta(X - \{x\})}(E)$

where

- E is the translation of φ
- $X = \textbf{free}(\varphi)$

Example

For φ with free variables x, y, z and translation E,
$\exists y\ \varphi$ is translated to $\pi_{A_x, A_z}(E)$

($\Leftarrow$) From calculus to algebra

Comparisons

x **op** y is translated to $\sigma_{\eta(x)\,\mathbf{op}\,\eta(y)}\left(\mathbf{Adom}_{\eta(x)} \times \mathbf{Adom}_{\eta(y)}\right)$

x **op** c is translated to $\sigma_{\eta(x)\,\mathbf{op}\,c}\left(\mathbf{Adom}_{\eta(x)}\right)$

Example

$x = y$ is translated to $\sigma_{A_x = A_y}\left(\mathbf{Adom}_{A_x} \times \mathbf{Adom}_{A_y}\right)$

$x > 1$ is translated to $\sigma_{A_x > 1}\left(\mathbf{Adom}_{A_x}\right)$

($\Leftarrow$) From calculus to algebra

Negation

$\neg\varphi$ is translated into $\left(\bigtimes_{x \in \mathbf{free}(\varphi)} \mathbf{Adom}_{\eta(x)} \right) - E$

where E is the translation of φ

Example

For φ with free variables x, y and translation E

$\neg\varphi$ is translated to $\mathbf{Adom}_{A_x} \times \mathbf{Adom}_{A_y} - E$

($\Leftarrow$) From calculus to algebra

Disjunction: $\varphi_1 \vee \varphi_2$ is translated to

$$E_1 \times \Big(\bigtimes_{x \in X_2 - X_1} \mathbf{Adom}_{\eta(x)} \Big) \cup E_2 \times \Big(\bigtimes_{x \in X_1 - X_2} \mathbf{Adom}_{\eta(x)} \Big)$$

where, for $i \in \{1, 2\}$,

- E_i is the translation of φ_i
- $X_i = \mathbf{free}(\varphi_i)$

Conjunction: same as disjunction, but use $\cap$ instead of $\cup$

Example

Customer : CustID, Name
Account : Number, CustID

Translate $\exists x_4 \, (\text{Customer}(x_1, x_2) \wedge \text{Account}(x_3, x_4) \wedge x_1 = x_4)$

Environment $\eta = \{ \, x_1 \mapsto A, \, x_2 \mapsto B, \, x_3 \mapsto C, x_4 \mapsto D \, \}$

$$\pi_{A,B,C}\Big(\big(E_1 \times \mathbf{Adom}_C \times \mathbf{Adom}_D\big) \cap \big(\mathbf{Adom}_A \times \mathbf{Adom}_B \times E_2\big) \cap \big(\sigma_{A=D}(\mathbf{Adom}_A \times \mathbf{Adom}_D) \times \mathbf{Adom}_B \times \mathbf{Adom}_C\big)\Big)$$

where

- $E_1 = \rho_{\text{CustID} \to A, \, \text{Name} \to B}(\text{Customer})$
- $E_2 = \rho_{\text{Number} \to C, \, \text{CustID} \to D}(\text{Account})$