Chapter 8
Relational Database Languages: Relational Calculus

Overview

- Described up to now: relational algebra, SQL
- the relational calculus is a specialization of the first-order calculus, tailored to relational databases.
- straightforward: the only structuring means of relational databases are relations – each relation can be seen as an interpretation of a predicate.
- there exists a **declarative** semantics.

8.1 First-Order Logic and the Relational Calculus

The relational calculus is a specialization of first-order logic.
(This section can be skipped or compressed depending on the knowledge of the participants)

8.1.1 Syntax

- **first-order language** contains a set of distinguished symbols:
  - "(" and ")", logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, quantifiers $\forall$, $\exists$,
  - an infinite set of variables $X, Y, X_1, X_2, \ldots$.
- An individual first-order language is then given by its **signature** Σ. Σ contains **function symbols** and **predicate symbols**, each of them with a given arity.

For databases:

- the relation names are the predicate symbols (with arity), e.g. $continent/2$, $encompasses/3$, etc.
- there are only 0-ary function symbols, i.e., **constants**.
- thus, the database schema $\mathbf{R}$ is the signature.

Syntax (Cont'd).

Terms

The set of **terms** over Σ is defined inductively as

- each variable is a term,
- for every function symbol $f \in \Sigma$ with arity n and terms $t_1, \ldots, t_n$, also $f(t_1, \ldots, t_n)$ is a term.

  0-ary function symbols: c, 1,2,3,4, "Berlin",...

  Example: for $plus/2$, the following are terms: $plus(3,4)$, $plus(plus(1,2),4)$, $plus(X,2)$.
- **ground terms** are terms without variables.

For databases:

- since there are no function symbols,
- the only terms are the **constants** and **variables**
  e.g., 1, 2, "D", "Germany", X, Y, etc.

Syntax (Cont'd): Formulas

Formulas are built inductively (using the above-mentioned special symbols) as follows:

Atomic Formulas

(1) For a predicate symbol (i.e., a relation name) R of arity k, and terms $t_1, \ldots, t_k$, $R(t_1, \ldots, t_k)$ is a formula.

(2) (**for databases only, as special predicates**)
A **selection condition** is an expression of the form $t_1 \, \theta \, t_2$ where t_1, t_2 are terms, and θ is a comparison operator in $\{=, \neq, \leq, <, \geq, >\}$.
Every selection condition is a formula.

(both are also called **positive literals**)

For databases:

- the atomic formulas are the **predicates** built over relation names and these constants, e.g.,
  continent("Asia",4.5E7), encompasses("R","Asia",X), country(N,CC,Cap,Prov,Pop,A).
- comparison predicates (i.e., the "selection conditions") are atomic formulas, e.g.,
  $X =$ "Asia", $Y > 10.000.000$ etc.

Syntax (Cont'd).

Compound Formulas

(3) For a formula F, also $\neg F$ is a formula. If F is an atom, $\neg F$ is called a **negative literal**.

(4) For a variable X and a formula F, $\forall X : F$ and $\exists F : X$ are formulas. F is called the **scope** of $\exists$ or $\forall$, respectively.

(5) For formulas F and G , the **conjunction** $F \wedge G$ and the **disjunction** $F \vee G$ are formulas.

For formulas F and G, where G (regarded as a string) is contained in F, G is a **subformula** of F.

The usual priority rules apply (allowing to omit some parentheses).

- instead of $F \vee \neg G$, the **implication** syntax $F \leftarrow G$ or $G \rightarrow F$ can be used, and
- $(F \rightarrow G) \wedge (F \leftarrow G)$ is denoted by the **equivalence** $F \leftrightarrow G$.

Syntax (Cont'd).

Bound and Free Variables

An occurrence of a variable X in a formula is

- **bound** (by a quantifier) if the occurrence is in a formula A inside $\exists X : A$ or $\forall X : A$ (i.e., in the scope of an appropriate quantifier).
- **free** otherwise, i.e.,if it is not bound by any quantifier.

Formulas without free variables are called **closed**.

Example:

- $continent($"Asia"$, X)$: X is free.
- $continent($"Asia"$, X) \wedge X > 10.000.000$: X is free.
- $\exists X : (continent($"Asia"$, X) \wedge X > 10.000.000)$: X is bound.
  The formula is closed.
- $\exists X : (continent(X, Y))$: X is bound, Y is free.
- $\forall Y : (\exists X : (continent(X, Y)))$: X and Y are bound.
  The formula is closed.

Outlook:

- closed formulas either hold in a database state, or they do not hold.
- free variables represent answers to queries:
  ?- $continent(\text{"Asia"}, X)$ means "for which value x does $continent(\text{"Asia"}, x)$ hold?"
  Answer: for $x = 4.5E7$.
- $\exists Y : (continent(X, Y))$: means
  "for which values x is there an y such that $continent(x, y)$ holds? – we are not interested in the value of y"
  The answer are all names of continents, i.e., that x can be "Asia", "Europe", or ...

... so we have to **evaluate** formulas ("semantics").

8.1.2 Semantics

The semantics of first-order logic is given by **first-order structures** over the signature:

First-Order Structure

A **first-order structure** $\mathcal{S} = (I, \mathcal{D})$ over a signature Σ consists of a nonempty set $\mathcal{D}$ (**domain**) and an interpretation I of the signature symbols over $\mathcal{D}$ which maps

- every constant c to an element $I(c) \in \mathcal{D}$,
- every n-ary function symbol f to an n-ary function $I(f) : \mathcal{D}^n \to \mathcal{D}$,
- every n-ary predicate symbol p to an n-ary relation $I(p) \subseteq \mathcal{D}^n$.

For Databases:

- no function symbols with arity > 0

First-Order Structures: An Example

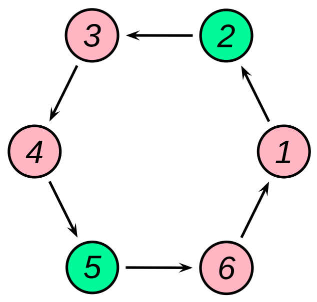
Example 8.1 (First-Order Structure)

Signature: *constant symbols:* $zero,\ one,\ two,\ three,\ four,\ five$

predicate symbols: $green/1,\ red/1,\ sees/2$

function symbols: $to\_right/1$, $plus/2$

Structure $\mathcal{S}$:



Domain $\mathcal{D} = \{0,\ 1,\ 2,\ 3,\ 4,\ 5\}$

Interpretation of the signature:

$I(zero) = 0,\ I(one) = 1, \ldots, I(five) = 5$

$I(green) = \{(2),\ (5)\},\ \ I(red) = \{(0),\ (1),\ (3),\ (4)\}$

$I(sees) = \{(0,3),\ (1,4),\ (2,5),\ (3,0),\ (4,1),\ (5,2)\}$

$I(to\_right) = \{\ (0) \mapsto (1),\ (1) \mapsto (2),\ (2) \mapsto (3),$
$(3) \mapsto (4),\ (4) \mapsto (5),\ (5) \mapsto (0)\}$

$I(plus) = \{(n,m) \mapsto (n+m)$ mod $6 \mid n,m \in \mathcal{D}\}$

Terms: $one,\ to\_right(four),\ to\_right(to\_right(X)),\ to\_right(to\_right(to\_right(four))),$
$plus(X, to\_right(zero)),\ to\_right(plus(to\_right(four), five))$

Atomic Formulas: $green(1),\ red(to\_right(to\_right(to\_right(four)))),\ sees(X,Y),$
$sees(X, to\_right(Z))$, $sees(to\_right(to\_right(four)), to\_right(one))$,
$plus(to\_right(to\_right(four)), to\_right(one)) = to\_right(three)$ □

Summary: Notions for Databases

- a set $\mathbf{R}$ of relational schemata; logically spoken, $\mathbf{R}$ is the **signature**,
- a database state is a structure $\mathcal{S}$ over $\mathbf{R}$
- $\mathcal{D}$ contains all domains of attributes of the relation schemata,
- for every single relation schema $R = (\bar{X})$ where $\bar{X} = \{A_1, \ldots, A_k\}$, we write $R[A_1, \ldots, A_k]$. k is the **arity** of the relation name R.
- relation names are the predicate symbols. They are interpreted by relations, e.g., $I(encompasses)$
  (which we also write as $\mathcal{S}(encompasses)$).

For Databases:

- no function symbols with arity > 0
- constants are interpreted "by themselves":
  $I(4) = 4,\ \ I($"Asia"$) =$ "Asia"
- care for domains of attributes.

Evaluation of Terms and Formulas

Terms and formulas must be **evaluated** under a given interpretation – i.e., wrt. a given database state $\mathcal{S}$.

- Terms can contain variables.
- variables are not interpreted by $\mathcal{S}$.

A **variable assignment** over a universe $\mathcal{D}$ is a mapping

$$\beta : Variables \to \mathcal{D} \ .$$

For a variable assignment β, a variable X, and $d \in \mathcal{D}$, the **modified** variable assignment β_X^d is identical with β except that it assigns d to the variable X:

$$\beta_X^d = \begin{cases} Y \mapsto \beta(Y) & \text{for } Y \neq X \ , \\ X \mapsto d & \text{otherwise.} \end{cases}$$

Example 8.2
For variables X, Y, Z, $\beta = \{X \mapsto 1,\ Y \mapsto$ "Asia"$, Z \mapsto 3.14\}$ is a variable assignment.

$\beta_X^3 = \{X \mapsto 3,\ Y \mapsto$ *"Asia"*$, Z \mapsto 3.14\}$. □

Evaluation of Terms

Terms and formulas are interpreted

- under a given interpretation $\mathcal{S}$, and
- wrt. a given variable assignment β.

For Databases:

- $\mathcal{S}$ is a database state.
- Σ is a purely relational signature,
- no function symbols with arity > 0, no nontrivial terms,
- constants are interpreted "by themselves".

Every interpretation $\mathcal{S}$ together with a variable assignment β induces an evaluation $\mathcal{S}$ of terms ($\mathcal{S}(t, \beta) \in \mathcal{D}$) and tuples of terms:

For Databases: $\mathcal{S}(x, \beta) := \beta(x)$ for a variable x ,
$\mathcal{S}(c, \beta) := c$ for a constant c .

Evaluation of Terms

Relevant only for full first-order logic:

$$\mathcal{S}(x, \beta) := \beta(x) \quad \text{for a variable } x\,,$$
$$\mathcal{S}(f(t_1, \ldots, t_n), \beta) := (I(f))(\mathcal{S}(t_1, \beta), \ldots, \mathcal{S}(t_n, \beta))$$

for a function symbol $f \in \Sigma$ with arity n and terms $t_1, \ldots, t_n$.

Example 8.3 (Evaluation of Terms)
Consider again Example 8.1.

- *For variable-free terms:* $\beta = \emptyset$.
- $\mathcal{S}(one, \emptyset) = I(one) = 1$
- $\mathcal{S}(to\_right(four), \emptyset) = I(to\_right(\mathcal{S}(four, \emptyset)) = I(to\_right(4)) = 5$
- $\mathcal{S}(to\_right(to\_right(to\_right(four))), \emptyset) = I(to\_right(\mathcal{S}(to\_right(to\_right(four)), \emptyset))) =$
  $I(to\_right(I(to\_right(\mathcal{S}(to\_right(four), \emptyset))))) = I(to\_right(I(to\_right(5)))) =$
  $I(to\_right(6)) = 1$ □

Example 8.3 (Continued)

- *Let* $\beta = \{X \mapsto 3\}$.
  $\mathcal{S}(to\_right(to\_right(X)), \beta) = I(to\_right(\mathcal{S}(to\_right(X), \beta))) =$
  $I(to\_right(I(to\_right(\mathcal{S}(X, \beta))))) = I(to\_right(I(to\_right(\beta(X))))) =$
  $I(to\_right(I(to\_right(3)))) = I(to\_right(4)) = 5$
- *Let* $\beta = \{X \mapsto 3\}$.
  $\mathcal{S}(plus(X, to\_right(zero)), \emptyset) = I(plus(\mathcal{S}(X, \beta), \mathcal{S}(to\_right(zero), \beta))) =$
  $I(plus(\beta(X), I(to\_right(\mathcal{S}(zero, \beta))))) = I(plus(3, I(to\_right(I(zero))))) =$
  $I(plus(3, I(to\_right(0)))) = I(plus(3, 1)) = 4$ □

EVALUATION OF FORMULAS

Formulas can either hold, or not hold in a database state.

Truth Value

Let F a formula, $\mathcal{S}$ an interpretation, and β a variable assignment of the free variables in F (denoted by $free(F)$).

Then we write $\mathcal{S} \models_\beta F$ if "F is true in $\mathcal{S}$ wrt. β".

Formally, $\models$ is defined inductively.

TRUTH VALUES OF FORMULAS: INDUCTIVE DEFINITION

Motivation: variable-free atoms

For an atom $R(a_1, \ldots, a_k)$, where a_i, $1 \leq i \leq k$ are constants,

$R(a_1, \ldots, a_k)$ is **true** in $\mathcal{S}$ if and only if $(I(a_1), \ldots, I(a_k)) \in \mathcal{S}(R)$.

Otherwise, $R(a_1, \ldots, a_k)$ is **false** in $\mathcal{S}$.

Base Case: Atomic Formulas

The **truth value** of an atom $R(t_1, \ldots, t_k)$, where t_i, $1 \leq i \leq k$ are terms, is given as

$\mathcal{S} \models_\beta R(t_1, \ldots, t_k)$ if and only if $(\mathcal{S}(t_1), \ldots, \mathcal{S}(t_k)) \in \mathcal{S}(R)$.

For Databases:

- the t_i can only be constants or variables.

Truth Values of Formulas: Inductive Definition

(2) $t_1 \, \theta \, t_2$ with θ a comparison operator in {=,$\neq$,$\leq$,$<$,$\geq$,$>$}:
$\mathcal{S} \models_\beta t_1 \, \theta \, t_2$ if and only if $\mathcal{S}(t_1, \beta) \; \theta \; \mathcal{S}(t_2, \beta)$ holds.

(3) $\mathcal{S} \models_\beta \neg G$ if and only if $\mathcal{S} \not\models_\beta G$.

(4) $\mathcal{S} \models_\beta G \wedge H$ if and only if $\mathcal{S} \models_\beta G$ and $\mathcal{S} \models_\beta H$.

(5) $\mathcal{S} \models_\beta G \vee H$ if and only if $\mathcal{S} \models_\beta G$ or $\mathcal{S} \models_\beta H$.

(6) $\mathcal{S} \models_\beta \forall X G$ if and only if for all $d \in \mathcal{D}$, $\mathcal{S} \models_{\beta_X^d} G$.

(7) $\mathcal{S} \models_\beta \exists X G$ if and only if for some $d \in \mathcal{D}$, $\mathcal{S} \models_{\beta_X^d} G$.

Example 8.4 (Evaluation of Atomic Formulas)
Consider again Example 8.1.

- *For variable-free formulas, let* $\beta = \emptyset$
- $\mathcal{S} \models_\emptyset green(1) \;\Leftrightarrow\; (1) \in I(green)$ – *which is not the case. Thus,* $\mathcal{S} \not\models_\emptyset green(1)$.
- $\mathcal{S} \models_\emptyset red(to\_right(to\_right(to\_right(four)))) \;\Leftrightarrow$
  $(\mathcal{S}(to\_right(to\_right(to\_right(four))), \emptyset)) \in I(red) \;\Leftrightarrow\; (6) \in I(red)$
  which is the case. Thus, $\mathcal{S} \models_\emptyset red(to\_right(to\_right(to\_right(four))))$.
- *Let* $\beta = \{X \mapsto 3, \; Y \mapsto 5\}$.
  $\mathcal{S} \models_\beta sees(X, Y) \;\Leftrightarrow\; (\mathcal{S}(X, \beta), \mathcal{S}(Y, \beta)) \in I(sees) \;\Leftrightarrow\; (3, 5) \in I(sees)$
  which is not the case.
- *Again,* $\beta = \{X \mapsto 3, \; Y \mapsto 5\}$.
  $\mathcal{S} \models_\beta sees(X, to\_right(Y)) \;\Leftrightarrow\; (\mathcal{S}(X, \beta), \mathcal{S}(to\_right(Y), \beta)) \in I(sees) \;\Leftrightarrow\; (3, 6) \in I(sees)$
  which is the case.
- $\mathcal{S} \models_\beta plus(to\_right(to\_right(four)), to\_right(one)) = to\_right(three) \;\Leftrightarrow$
  $\mathcal{S}(plus(to\_right(to\_right(four)), to\_right(one)), \emptyset) = \mathcal{S}(to\_right(three), \emptyset) \;\Leftrightarrow\; 2 = 4$
  which is not the case. □

Example 8.5 (Evaluation of Compound Formulas)
Consider again Example 8.1.

- $\mathcal{S} \models_\emptyset \exists X : red(X) \;\Leftrightarrow$

  there is a $d \in \mathcal{D}$ *such that* $\mathcal{S} \models_{\emptyset^d_X} red(X) \;\Leftrightarrow$ *there is a* $d \in \mathcal{D}$ *s.t.* $\mathcal{S} \models_{\{X \mapsto d\}} red(X)$

  Since we have shown above that $\mathcal{S} \models_\emptyset red(6)$*, this is the case.*

- $\mathcal{S} \models_\emptyset \forall X : green(X) \;\Leftrightarrow$

  for all $d \in \mathcal{D}$, $\mathcal{S} \models_{\emptyset^d_X} green(X) \;\Leftrightarrow$ *for all* $d \in \mathcal{D}$, $\mathcal{S} \models_{\{X \mapsto d\}} green(X)$

  Since we have shown above that $\mathcal{S} \not\models_\emptyset green(1)$ *this is not the case.*

- $\mathcal{S} \models_\emptyset \forall X : (green(X) \vee red(X)) \;\Leftrightarrow$ *for all* $d \in \mathcal{D}$, $\mathcal{S} \models_{\{X \mapsto d\}} (green(X) \vee red(X))$.

  One has now to check whether $\mathcal{S} \models_{\{X \mapsto d\}} (green(X) \vee red(X))$ *for all* $d \in domain$.
  We do it for $d = 3$:

  $\mathcal{S} \models_{\{X \mapsto 3\}} (green(X) \vee red(X)) \;\Leftrightarrow$

  $\mathcal{S} \models_{\{X \mapsto 3\}} green(X)$ *or* $\mathcal{S} \models_{\{X \mapsto 3\}} red(X) \;\Leftrightarrow$

  $(\mathcal{S}(X, \{X \mapsto 3\})) \in I(green)$ *or* $(\mathcal{S}(X, \{X \mapsto 3\})) \in I(red) \;\Leftrightarrow$

  $(3) \in I(green)$ *or* $(3) \in I(red)$

  which is the case since $(3) \in I(red)$.

- *Similarly,* $\mathcal{S} \not\models_\emptyset \forall X : (green(X) \wedge red(X))$ □

8.2 Formulas as Queries

Formulas can be seen as **queries**:

- For a formula F with free variables $X_1, \ldots, X_n$, $n \geq 1$, we write $F(X_1, \ldots, X_n)$.

- each formula $F(X_1, \ldots, X_n)$ defines – dependent on a given interpretation $\mathcal{S}$ – an **answer relation** $\mathcal{S}(F(X_1, \ldots, X_n))$.

  The **answer set** to $F(X_1, \ldots, X_n)$ wrt. $\mathcal{S}$ is the set of tuples $(a_1, \ldots, a_n)$, $a_i \in \mathcal{D}$, $1 \leq i \leq n$, such that F is true in $\mathcal{S}$ when assigning each of the variables X_i to the constant a_i, $1 \leq i \leq n$.

  Formally:

  $$\mathcal{S}(F) = \{(\beta(X_1), \ldots, \beta(X_n)) \mid \mathcal{S} \models_\beta F \text{ where } \beta \text{ is a variable assignment of } free(F)\}.$$

- for $n = 0$, the answer to F is **true** if $\mathcal{S} \models_\emptyset F$ for the empty variable assignment $\emptyset$;
  the answer to F is **false** if $\mathcal{S} \not\models_\emptyset F$ for the empty variable assignment $\emptyset$.

Example 8.6

Consider the MONDIAL *schema.*

- *Which cities (CName, Country) have at least 1.000.000 inhabitants?*

$$F(CN, C) = \quad \exists\, Pr, Pop, L1, L2\ (\textit{city}(CN, C, Pr, Pop, L1, L2) \wedge\ Pop \geq 1000000)$$

- *Which countries (CName) belong to Europe?*

$$\begin{aligned} F(CName) = \exists\, & CCode, Cap, Capprov, Pop, A, ContName, ContArea \\ & (\textit{country}(CName, CCode, Cap, Capprov, Pop, A) \wedge \\ & \ \textit{continent}(ContName, ContArea) \wedge \\ & \ ContName = \text{'Europe'} \ \wedge\ \textit{encompasses}(ContName, CCode)\ ) \end{aligned}$$

□

Example 8.6 (Continued)

- *Again, relational division ...*
  Which organizations have at least one member on each continent

$$\begin{aligned} F(Abbrev) = \exists O, & HeadqN, HeadqC, HeadqP, Est : \\ & (organization(O, Abbrev, HeadqN, HeadqC, HeadqP, Est) \wedge \\ & \ \forall Cont :\ ((\exists ContArea :\ continent(Cont, ContArea)) \rightarrow \\ & \qquad \exists Country, Perc, Type : (encompasses(Country, Cont, Perc) \wedge \\ & \qquad\qquad isMember(Country, Abbrev, Type)))) \end{aligned}$$

- *Negation*
  All pairs (country,organization) such that the country is a member in the organization, and all its neighbors are not.

$$\begin{aligned} F(CCode, Org) = \exists & CName, Cap, Capprov, Pop, Area, Type : \\ & (\textit{country}(CName, CCode, Cap, Capprov, Pop, Area) \wedge \\ & \ \textit{isMember}(CCode, Org, Type) \wedge \\ & \ \forall CCode' : (\exists Length : \textit{sym\_borders}(CCode, CCode', Length) \rightarrow \\ & \qquad \neg\exists Type' : \textit{isMember}(CCode', Org, Type'))) \end{aligned}$$

□

8.3 Comparison of the Algebra and the Calculus

Calculus: The semantics (= answer) of a query in the relational calculus is defined via the truth value of a formula wrt. an interpretation

"**declarative** Semantics".

Algebra: The semantics is given by evaluating an algebraic expression (i.e., an operator tree)

"**algebraic** Semantics".

Example: Expressing Algebra Operations in the Calculus

Consider relation schemata $R[A,B]$, $S[B,C]$, and $T[A]$.

- **Projection** $\pi[A]R$:
$$F(X) = \exists Y\, R(X,Y)$$
- **Selection** $\sigma[A=B]R$:
$$F(X,Y) = R(X,Y) \wedge X = Y$$
- **Join** $R \bowtie S$:
$$F(X,Y,Z) = R(X,Y) \wedge S(Y,Z)$$
- **Union** $R \cup (T \times \{b\})$:
$$F(X,Y) = R(X,Y) \vee (T(X) \wedge Y = b)$$
- **Difference** $R - (T \times \{b\})$:
$$F(X,Y) = R(X,Y) \wedge \neg(T(X) \wedge Y = b)$$
- **Division** $R \div T$:
$$F(Y) = \forall X : (T(X) \Rightarrow R(X,Y)) \qquad \text{or} \qquad F(X) = \neg\exists X : (T(X) \wedge \neg R(X,Y))$$

SAFETY AND DOMAIN-INDEPENDENCE

- If the domain $\mathcal{D}$ is infinite, the answer relations to some expressions of the calculus can be infinite!

Example 8.7
Let

$$F(X) = \neg R(X),$$

("give me all a such that $R(a)$ does not hold")

where $\mathcal{S}(R) = \{1\}$.

Depending on $\mathcal{D}$, $\mathcal{S}(F)$ is infinite. □

Example 8.8
Let

$$F(X, Z) = \exists Y(R(X, Y) \vee S(Y, Z)),$$

Consider $\mathcal{S}(R) = \{(1, 1)\}$, arbitrary $\mathcal{S}(S)$ (even empty).

Which Z? □

Example 8.9
Consider a database of persons:

married(X,Y): X is married with Y.

$F(X) = \neg married(john, X) \wedge \not(X = john).$

What is the answer?

- *Consider* $\mathcal{D} = \{john, mary\}$, $\mathcal{S}(married) = \{(john, mary), (mary, john)\}$.
  $\mathcal{S}(F) = \emptyset$.
  - *there is no person (except John) who is not married with John*
  - *all persons are married with John???* □
- *Consider* $\mathcal{D} = \{john, mary, sue\}$, $\mathcal{S}(married) = \{(john, mary), (mary, john)\}$.
  $\mathcal{S}(F) = \{sue\}$.

  The answer depends not only on the database, but on the domain (that is a purely logical *notion)*

  Obviously, it is meant "All persons in the database who are not married with $john$".

Active Domain

Requirement: the answer to a query depends only on

- constants given in the query
- constants in the database

Definition 8.1
Given a formula F of the relational calculus and a database state $\mathcal{S}$, $DOM(F)$ contains

- *all constants in F,*
- *and all constants in $\mathcal{S}(R)$ where R is a relation name that occurs in F.*

*$DOM(F)$ is called the **active domain** domain of F.* □

$DOM(F)$ is finite.

Domain-Independence

Formulas in the relational calculus are required to be **domain-independent**:

Definition 8.2
*A formula $F(X_1, \ldots, X_n)$ is **domain-independent** if for all $D \supseteq DOM(F)$,*

$$\begin{aligned}\mathcal{S}(F) &= \{(\beta(X_1), \ldots, \beta(X_n)) \mid \mathcal{S} \models_\beta F,\ \beta(X_i) \in DOM(F) \text{ for all } 1 \leq i \leq n\} \\ &= \{(\beta(X_1), \ldots, \beta(X_n)) \mid \mathcal{S} \models_\beta F,\ \beta(X_i) \in D \text{ for all } 1 \leq i \leq n\}.\end{aligned}$$ □

It is undecidable whether a formula F is domain-independent!
(follows from Rice's Theorem).

Instead, **(syntactical) safety** is required for queries:

- stronger condition
- can be tested algorithmically

Safety

Definition 8.3
A formula F is ***(syntactically) safe*** *if and only if it satisfies the following conditions:*

1. *F does not contain $\forall$ quantifiers. (for formal simplicity since $\forall X G$ can always be replaced by $\neg \exists X \neg G$)*
2. *if $F_1 \vee F_2$ is a subformula of F, then F_1 and F_2 must have the same free variables.*
3. *for all maximal conjunctive subformulas $F_1 \wedge \ldots \wedge F_m, m \geq 1$ of F:*
   All free variables must be ***bounded****:*
   - *Let $1 \leq j \leq m$.*
   - *if F_j is neither a comparison, nor a negated formula, any free variable in F_j is bounded,*
   - *if F_j is of the form $X = a$ or $a = X$ with a a constant, then X is bounded,*
   - *if F_j is of the form $X = Y$ or $Y = X$ and Y is bounded, then X is also bounded.*

(a subformula G of a formula F is a ***maximal conjunctive subformula****, if there is no conjunctive subformula H of F such that G is a subformula of H).* □

Example 8.10

- *$X = Y \vee R(X, Z)$ is not safe*
- *$X = Y \wedge R(X, Y)$ is safe*
- *$R(X, Y, Z) \wedge \neg(S(X, Y) \vee T(Y, Z))$ is not safe, but the logically equivalent formula*

$$R(X, Y, Z) \wedge \neg S(X, Y) \wedge \neg T(Y, Z)$$

  is safe. □

- safety is defined purely syntactically
- safety can be tested effectively
- safety implies domain-independence
  (proof by induction on the number of maximal conjunctive subformulas).

8.4 Equivalence of Algebra and (safe) Calculus

As for the algebra, the attributes of each relation are assumed to be ordered.

Theorem 8.1

For each expression Q of the relational algebra there is an equivalent safe formula F of the relational calculus, and vice versa; i.e., for every state $\mathcal{S}$, Q and F define the same answer relation. □

Proof:

(A) Algebra to Calculus

Let Q an expression of the relational algebra. The proof is done by induction over the structure of Q (as an operator tree). The formulas that are generated are always safe.

Induction base: Q does not contain operators.

- if $Q = R$ where R is a relation symbol of arity $n \geq 1$:

$$F(Z_1, \ldots, Z_n) = R(Z_1, \ldots, Z_n)$$

R	
A_1	A_2
a	b
1	2

Q: R answer to $R(Z_1, Z_2)$:

Z_1	Z_2
a	b
1	2

- otherwise, $Q = \{c\}$, $c \in \mathcal{D}$. Then, $F(Z) = (Z = c)$.

{c}
?
c

Answer to $Z = c$:

Z
c

Induction step:

Assume that Q_1 is equivalent to $F_1(X_1, \ldots, X_m)$ and Q_2 is equivalent to $F_2(Y_1, \ldots, Y_n)$.

- Case $Q = Q_1 \cup Q_2$ where $\Sigma_{Q_1} = \Sigma_{Q_2}$ and $|\Sigma_{Q_1}| = n \geq 1$.

$$\begin{array}{lcll} F(Z_1, \ldots, Z_n) & = & \exists\ X_1, \ldots, \exists X_n & (F_1(X_1, \ldots, X_n) \wedge Z_1 = X_1 \wedge \ldots \wedge Z_n = X_n) \vee \\ & & \exists\ Y_1, \ldots, \exists Y_n & (F_2(Y_1, \ldots, Y_n) \wedge Z_1 = Y_1 \wedge \ldots \wedge Z_n = Y_n). \end{array}$$

Example:

Q_1	
A_1	A_2
a	b
c	d

$F_1($

X_1	X_2
a	b
c	d

$)$

Q_2	
A_1	A_2
1	2
c	d

$F_2($

Y_1	Y_2
1	2
c	d

$)$

$F($

Z_1	Z_2
a	b
c	d
1	2

$)$

- Case $Q = Q_1 - Q_2$. The same, replace $\ldots \vee \ldots$ by $\ldots \wedge \neg(\ldots)$.
- Case $Q = \pi[Y]Q_1$ and $Y = \{A_{i_1}, \ldots, A_{i_k}\} \subseteq \Sigma_{Q_1}$, $k \geq 1$.

$$F(Z_1, \ldots, Z_k) = \exists X_1, \ldots, \exists X_n (F_1(X_1, \ldots, X_n) \wedge Z_1 = X_{i_1} \wedge \ldots \wedge Z_k = X_{i_k}).$$

Example:

Q_1	
A_1	A_2
a	b
c	d

$F_1($

X_1	X_2
a	b
c	d

$)$

Let $Y = \{A_2\}$: $\quad F(Z_1) = \exists X_1, \exists X_2 (F_1(X_1, X_2) \wedge Z_1 = X_2)$

$F($

Z_1
b
d

$)$

- Case $Q = \sigma[\alpha]Q_1$, $A_i, A_j \in \Sigma_{Q_1}$ and $n \geq 1$.

$$F(X_1, \ldots, X_n) = F_1(X_1, \ldots, X_n) \wedge \alpha', \text{ where } \alpha' = \begin{cases} X_i \,\theta\, a_i & \text{for} \quad \alpha = (A_i \,\theta\, a_i), \\ a_i \,\theta\, X_i & \text{for} \quad \alpha = (a_i \,\theta\, A_i), \\ X_i \,\theta\, X_j & \text{for} \quad \alpha = (A_i \,\theta\, A_j). \end{cases}$$

Example:

Q_1	
A_1	A_2
1	2
3	4

$F_1($

X_1	X_2
1	2
3	4

$)$

Let $\sigma =$ "$A_1 = 3$": $\quad F(Z_1, Z_2) = F_1(X_1, X_2) \wedge Z_1 = 3$

$F($

Z_1	Z_2
3	4

$)$

- Case $Q = Q_1 \bowtie Q_2$ and $\Sigma_{Q_1} = \{A_1, \ldots, A_m\}$, $\Sigma_{Q_2} = \{B_1, \ldots, B_n\}$, $n, m \geq 1$. Let w.l.o.g. $A_1 = B_1, \ldots, A_k = B_k$ for some $k \leq n, m$.

$$F(X_1, \ldots, X_m, Y_{k+1}, \ldots, Y_n) = (F_1(X_1, \ldots, X_m) \wedge F_2(Y_1, \ldots, Y_n) \wedge \\ \wedge X_1 = Y_1 \wedge \ldots \wedge X_k = Y_k).$$

Example:

Q_1	
AB_1	A_2
1	2
3	4

Q_2	
AB_1	B_2
5	6
1	7

$F_1($

X_1	X_2
1	2
3	4

$)$

$F_2($

Y_1	Y_2
5	6
1	7

$)$

$F(Z_1, Z_2, Z_3) = F_1(X_1, X_2) \wedge F_2(Y_1, Y_2) \wedge X_1 = Y_1$

$F($

Z_1	Z_2	Z_3
1	2	7

$)$

Note again that the resulting formulas F are safe.

(B) Calculus to Algebra

Consider a safe formula $F(X_1, \ldots, X_n)$, $n \geq 1$ of the relational calculus.

First, an algebra expression E that computes the active domain $DOM(F)$ of the formula and the database is derived:

Assume $R_1, \ldots, R_n$, $n \geq 0$ to be the relation names in F. For k-ary R_i,

$$E(R_i) = \pi[\$1](R_i) \cup \ldots \cup \pi[\$k](R_i).$$

Let

$$E = E(R_1) \cup \ldots E(R_n) \cup \{a_1, \ldots, a_m\},$$

where $a_j, 1 \leq j \leq m$ are the constants in F.

- $E(\mathcal{S})$ is a unary relation.

An equivalent algebra expression Q is now constructed by induction over the number of maximal conjunctive subformulas of F.

Induction base: F has exactly one maximal conjunctive subformula. Thus, $F = G_1 \wedge \ldots \wedge G_l$, $l \geq 1$.

(1) Case $l = 1$.
Then, either $F = R(a_1, \ldots, a_k)$, where a_i are variablen or constants, or F is a comparison of one of the forms $F = (X = a)$ or $F = (a = X)$, where X is a variable and a is a constante (note that all other comparisons would not be safe).

- Case $F = R(a_1, \ldots, a_k)$, e.g. $F = R(a, X, b, Y, a, X)$. Then, let

$$Q = \pi[\$2, \$4](\sigma[\Theta_1 \wedge \Theta_2](R)) \ ,$$

where

$$\Theta_1 = (\$1 = a \wedge \$3 = b \wedge \$5 = a)$$

and

$$\Theta_2 = (\$2 = \$6)$$

- Case $F = (X = a)$ or $F = (a = X)$. Let

$$Q = \{a\} \ .$$

(2) Case $l > 1$ (cf. example below) Then, w.l.o.g.

$$F = G_1 \wedge \ldots \wedge G_u \wedge G_{u+1} \wedge \ldots \wedge G_v$$

s.t. $u + v > 1$, where all G_i, $1 \leq i \leq u$ as in (1) and all G_j, $u < j \leq v$ are other comparisons.

For every G_i, $1 \leq i \leq u$ take an algebra expression $Q(G_i)$ as done in (1), where the format $\Sigma_{Q(G_i)}$ is just the set of free variables in G_i. Let

$$Q' = \bowtie_{i=1}^{u} Q(G_i).$$

With Θ the conjunction of the selection conditions $G_{u+1}, \ldots, G_v$,

$$Q = \sigma[\Theta]Q' \; .$$

Example 8.11

Consider $F = R(a, X, b, Y, a, X) \wedge S(X, Z, a) \wedge X = Y$
as $F = G_1 \wedge G_2 \wedge G_3$*:*

$$Q(G_1) = \pi[\$2, \$4](\sigma[\$1 = a \wedge \$3 = b \wedge \$5 = a \wedge \$1 = \$6](R))$$
$$Q(G_2) = \pi[\$1, \$2](\sigma[\$3 = a](S))$$

$$Q(F) = \sigma[X = Y]((([\$1 \to X, \$2 \to Y]Q(G_1)) \bowtie ([\$1 \to X, \$2 \to Z]Q(G_2)))$$

□

Induction Step: For formulas $F, G, H, \ldots$ with maximal $n-1$ maximal conjunctive subformulas, the equivalent algebra expressions are $Q(F), Q(G), Q(H), \ldots$.

(3) $F = \exists X G$.

$$Q = \pi[\$1, \ldots, \$k](Q(G)) \; ,$$

where G has $k+1$, $k \geq 0$ free variables, and w.l.o.g. X is the $k+1$th free variable.

(4) $F = G \vee H$.

$$Q = Q(G) \cup Q(H)$$

(safety guarantees that G and H have the same free variables, thus, $Q(G)$ and $Q(H)$ have the same format).

(5) $F = G_1 \wedge \ldots \wedge G_l$, $l \geq 1$ where some G_i are of the form $\neg H_i$. Then,

$$Q(G_i) = E^k - Q(H_i)$$

where $Q(H_i)$ is k-ary.

Q is then constructed analogous to (2).