2.3.- Relational Data Model (logic approach)

There are two logic-based manipulation languages for the relational model:

- The Tuple Relational Calculus.
- The Domain Relational Calculus.

- The logic approach in the relational data model is useful to make *queries* and express *constraints*.

- The **Tuple Relational Calculus** is the main reference over which the manipulation language in SQL is constructed.

2.3.1.- First order logic

FIRST ORDER LOGIC

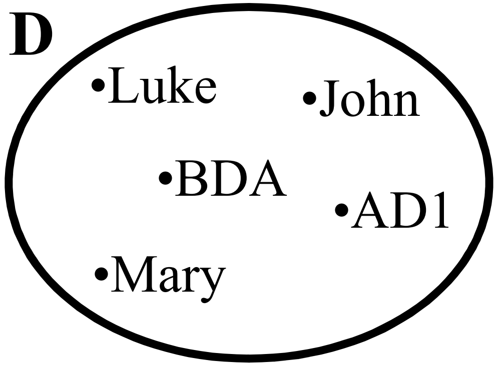
- "Formal system which makes it possible to reason over a *universe of discourse*"

- First order logic is a formal language. Hence, it has two elements:
    - A **language** in which we can express assertions over the universe of interest. (SYNTAX)
    - Several **rules** with which the truth value of any stated assertions can be determined. (SEMANTICS)

Example: "Mortal(Socrates)" and "Planet (Socrates)" are syntactically correct, but only the first one is semantically true in our world.

2.3.1.- First order logic

D



P:

- •is a student
- •is a course
- •is registered (a student in a course)

ASSERTION: "All the students are registered in some course"

Is this assertion true?

We need more information about the properties of ***P*** in domain ***D***

If this information is:

- is_a_student = {John, Luke, Mary}
- is_a_course = {AD1, BDA}
- is_ registered = {(Luke,AD1), (John,BDA)}

The assertion is <u>false</u> for the knowledge we have about the properties ***P*** in domain ***D***

2.3.1.- First order logic

FORMALISATION (SYNTAX):

- We must define a first-order language ***L*** to refer to the individuals and the properties of the universe of discourse:

  L:

  Constants = {Luke, Mary, John, AD1, BDA}

  Predicates = {Student(.), Course(.), Inscription(.,.)}

  Variables = {x, y}

  Connectives = {$\rightarrow$, $\neg$, $\wedge$, $\vee$}

  Quantifiers = {$\forall$, $\exists$ }

- Example of syntactically correct formulas:

  F: $\forall x\ (Student(x) \rightarrow \exists y\ Inscription(x, y))$

  G: Student(x) ^ Inscription(x,'AD1')

- Example of a syntactically incorrect formula:

  F': $\forall$Student $\exists$Inscription(x, y)

2.3.1.- First order logic

FORMALISATION (SEMANTICS):

- The interpretation ***I*** of the previous first-order language in the domain ***D*** *corresponding to the previous example*:

  D = {Luke, Mary, John, AD1, BDA}

  Student = {John, Luke, Mary}

  Course = {AD1, BDA}

  Registration = {(Luke, AD1), (John, BDA)}

- The evaluation of **F:** $\forall x\ (\text{Student}(x) \rightarrow \exists y\ \text{Registration}(x, y))$ in ***I*** is performed following the following fixed rules.

2.3.1.- First order logic

FORMULA EVALUATION IN FIRST-ORDER LOGIC (SEMANTICS).

Given:

- A formula F.
- An interpretation I.
- A domain D.
- An assignment of values to the free variables in F.

The rules for the evaluation of F are as follows:

2.3.1.- First order logic

1) If F is a comparison:
    - if F is of the form $X \alpha Y$ where X and Y are constants or variables, then F is evaluated to the truth value of the comparison.

2) If F is an n-ary predicate of the form $R(x_1, \ldots, x_n)$, then F is evaluated to <u>true</u> if $(x_1, \ldots, x_n)$ belongs to the interpretation of R in I; otherwise, F is evaluated to <u>false</u>.

3) If F is of the form (G), F is evaluated to the truth value of G.

2.3.1.- First order logic

4) If F is like one of the following expressions $\neg G$, $G \wedge H$, $G \vee H$ or $G \rightarrow H$ where G and H are well-formed formulas, then F is evaluated according to the following truth tables:

G	H	F = G ∧ H	F = G ∨ H	F = G → H
false	false	false	false	true
undefined	false	false	undefined	undefined
true	false	false	true	false
false	undefined	false	undefined	true
undefined	undefined	undefined	undefined	undefined
true	undefined	undefined	true	undefined
false	true	false	true	true
undefined	true	undefined	true	true
true	true	true	true	true

G	F = ¬ G
false	true
undefined	undefined
true	false

2.3.1.- First order logic

5) If F is of the form $\exists x\ G$, then F is true if there is an assignment for the variable x which makes the formula G true.

6) If F is of the form $\forall x\ G$, then F is true if for every assignment for the variable x, it makes the formula G true.

2.3.1.- First order logic

FORMALISATION (SEMANTICS):

EVALUATION OF OPEN AND CLOSED FORMULAE:

- Closed formulas are used to express assertions (constraints).
- Open formulas are used to express queries.

Given the previous interpretation example:

D = {Luke, Mary, John, AD1, BDA}

Student = {John, Luke, Mary}

Course = {AD1, BDA}

Registration = {(Luke, AD1), (John, BDA)}

A closed formula:

$\forall x$ (Student(x) $\rightarrow \exists y$ Registration(x, y)). ➡ All the variables are affected by a quantifier (bound variables)

An open formula:

Student(x) ^ Registration(x, 'AD1')). ➡ There are variables which are not affected by quantifiers (free variables)

2.3.1.- First order logic

EVALUATION OF A CLOSED FORMULA (SEMANTICS).

- The evaluation of **F:** $\forall x\ (\text{Student}(x) \rightarrow \exists y\ \text{Registration}(x, y))$.

Given the previous interpretation example:

D = {Luke, Mary, John, AD1, BDA}

Student = {John, Luke, Mary}

Course = {AD1, BDA}

Registration = {(Luke, AD1), (John, BDA)}

- Following the rules we have seen, we can deduce that:

"***F*** is false in ***I***"

2.3.1.- First order logic

EVALUATION OF AN OPEN FORMULA (SEMANTICS).

An open formula:

Student(x) ^ Registration(x, ‘AD1’)).

Given the previous interpretation example:

D = {Luke, Mary, John, AD1, BDA}
Student = {John, Luke, Mary}
Course = {AD1, BDA}
Registration = {(Luke, AD1), (John, BDA)}

EVALUATION:

- Look for values in the domain such that, assigned to the free variables (in this case *x*), make the formula true.

SOLUTION:

x = {‘Luke’}

2.3.2.- Logical interpretation of a relational database

- An **interpretation** consists of an association of each n-ary predicate with an n-ary relation defined over a domain ***D***:

Student

John
Luke
Mary

Course

AD1
BDA

Registration

Luke	AD1
John	BDA

- Thus, an interpretation in a first-order language can be seen as a *relational database* in which:
    - The names of the relation match the names of the predicates.
    - The domain of the attributes match the set of constants.

2.3.2.- Logical Interpretation of a relational database

Province

p_id	prov_name
44	Teruel
46	Valencia
16	Cuenca
12	Castellón

River

r_id	name
r1	Sénia
r2	Túria
r3	Xúquer

Is_crossed_by

p_id	r_id
44	r1
46	r2
30	r2
20	r1
44	r3
12	r1

Predicates: {Province(.,.) River(.,.) Is_crossed_by(.,.)}

Interpretation: The extensions of the relations in the database

F1: River(x,y) ^ Is_crossed_by(44,x) ⇒ x={'r1'} y={'Sénia'}, also x={'r3'} y={'Xúquer'}

F2: River(x,y) ^ ¬∃z Is_crossed_by(z,x) ⇒ x={} y={}

2.3.2.- Logical Interpretation of a relational database

Tuple variables:

- are variables which are declared over the database relations *Tuple_variable:relation_name*.
- their possible values are restricted to the tuples in the extension of the relation over which it was defined.
- their components can be referred to as follows *Tuple_variable.relation_attribute.*

EXAMPLE:

River(r_id:string(6), name:string(20))

RX : River

possible values for ***RX***:

{(r_id: “r1”), (name: “Sénia”)}
{(r_id: “r3”), (name: “Xúquer”)}
~~{(r_id: “xx”), (name: “xftrfsdh”)}~~
~~{(r_id: “r2”), (name: “Tajo”)}~~
~~{(r_id: “r3”), (name: “Túria”)}~~

id_r	**name**
r1	Sénia
r2	Túria
r3	Xúquer

2.3.2.- Logical Interpretation of a relational database

Queries with tuple variables:

- The queries with tuple variables have the following form:

$$\{Free\_variables\_declaration | Well\text{-}formed\_formula\}$$

- The examples written in first-order logic can be rewritten with tuple variables in the following way:

First order logic:

F1: River(x,y) ^ Is_crossed_by(44,x)

F2: River(x,y) ^ ¬∃z Is_crossed_by(z,x)

First order logic with tuple variables:

F1: RX:River | ∃PPX:Is_crossed_by (PPX.r_id = RX.r_id ^ PPX.p_id = 44)

F2: RX:River | ¬∃PPX:Is_crossed_by (RX.r_id = PPX.r_id)

2.3.2.- Logical Interpretation of a relational database

Queries with tuple variables:

EXAMPLES:

River(r_id:r_id_dom, name:name_dom, length:len_dom)

Province(p_id:p_id_dom, name:name_dom)

Is_crossed_by(p_id:p_id_dom, r_id:r_id_dom)

Which rivers cross at least two provinces?

RX:River | ∃PPX:Is_crossed_by, ∃PPY:Is_crossed_by

(RX.r_id = PPX.r_id ^ RX.r_id = PPY.r_id ^ PPX.p_id ≠ PPY.p_id)

Which rivers cross all provinces?

RX:River | ∀PX:Province (∃PPX:Is_crossed_by (RX.r_id = PPX.r_id ^ PPX.p_id = PX.p_id)

Or using equivalence ∀x F(x) ≡ ¬∃x (¬ F(x))

RX:River | ¬∃PX:Province (¬∃PPX:Is_crossed_by (RX.r_id = PPX.r_id ^ PPX.p_id = PX.p_id)