

Databases from a logical perspective

Ingmar Dasseville *ingmar.dasseville@cs.kuleuven.be*
Thomas Vanstrydonck *thomas.vanstrydonck@cs.kuleuven.be*
Simon Marynissen *simon.marynissen@cs.kuleuven.be*

October 17, 2017

In this exercise session we start from the concept of relational databases (in the rest of this assignment: database) and we investigate the aspects of databases from a logical perspective:

- Different logical visions for databases
- Modelling information in databases in logic
- The representation of queries on a database in logic as a modelling exercise.

1 A Database as a logical structure

Consider the next database (cf the course): From a logical perspective we

Instructor		Enrolled		Grade			PassGrade
Ray	CS230	Jill	CS230	Sam	CS148	AAA	
Hec	CS230	Jack	CS230	Bill	CS148	D	AAA
Sue	M100	Sam	CS230	Jill	CS148	A	AA
Sue	M200	Bill	CS230	Jack	CS148	C	A
Pat	CS238	May	CS238	Flo	CS230	AA	B
Prerequ ^a		Ann	CS238	May	CS230	AA	C
CS230	CS238	Tom	M100	Bill	CS230	F	
CS148	CS230	Ann	M100	Ann	CS230	C	
M100	M200	Jill	M200	Jill	M100	B	
		Sam	M200	Sam	M100	AA	
		Flo	M200	Flo	M100	D	
				Flo	M100	B	

^aPrerequ(a,b)
means a is a prerequisite for b

Figure 1: The student database

can directly see this database as a structure I over a vocabulary Σ .

$$\Sigma = \{Instructor/2, Grade/3, Enrolled/2, Prerequ/2, PassingGrade/1, \\ Ray, Hec, \dots, CS230, \dots, AAA, \dots, Jill, \dots\}$$

The domain is

$$Domain_I = \{Ray, Hec, \dots, CS230, \dots, AAA, \dots, Jill, \dots\}$$

The interpretation of I consists of

- The interpretation of all constants: $Ray^I = Ray, \dots, CS230^I = CS230, \dots, AAA^I = AAA, \dots$
- The relation of tuples of domain elements $Instructor^I, Grade^I, \dots$ as in Figure 1

I is the Herbrand structure.

Exercise 1. Formulate the next queries in Σ en evaluate them in structure I . All queries are *simple*: the answer is *true* or *false*. On Toledo you can find an example file in which you can fill the queries and use the IDP system to answer them.

1. *Is there a prerequisite for CS238?*
2. *Has May passed for CS230 ?*
3. *Are all the courses which are (direct) prerequisite for M100 instructed by Ray?*
4. *Has everyone that is enrolled in CS230 passed for at least one course instructed by Sue?*
5. *Did John do an exam for CS230?*
6. *Are there students taking a course from every instructor?*
7. *Is there are student who is enrolled in exactly 1 course?*

Exercise 2 An n -ary query is a symbolic definition of a relation of the form $\{(x_1, \dots, x_n) : A\}$. On Toledo you can find an example file in which you can write the queries and use the IDP system to answer them. Construct the following n -ary queries:

1. *All tuples (x, y) such that student x has passed for the course y .*
2. *All students who follow exactly one course*
3. *All students who are enrolled in CS230 and have passed for all the courses they did up til now.*

Exercise 3. An essential part of the database are the integrity constraints. The database is subject to these constraints. These need to be checked when the database is changed and need to prevent the creation of invalid relations. Express the following integrity constraints in Σ (on Toledo you can find an example file in which you can fill in the constraints and call the IDP system to test if they are satisfied in the given database).

1. *No subject is a (direct) prerequisite for itself*
2. *Nobody can have more than one grade for a course.*
3. *No student can be an instructor. (A student is someone who is enrolled in a course)*
4. *Everyone who is enrolled for a course needs to have passed all courses which are (direct) needed prerequisites for that course.*

2 SQL

The query language SQL is a widespread language for formulating queries on database systems. SQL is based on the principles of predicate logic but has a strongly dissimilar syntax. Nevertheless it isn't difficult to see the connection to logic.

In an SQL database the vocabulary (called the *database-scheme*) is given in the form of a number of table-declaration of the form:

$$table(field_1, field_2, \dots, field_n)$$

Here *table* is the name of the table and *field_i* is the name of the *i*th column of the table. E.g. *Grade(student, course, score)*, *Enrolled(student, score)*.

Exercise 4. Given this vocabulary we can translate SQL-queries to queries in logic. Translate the next SQL query to logic (you can do this on paper instead of in IDP)

```
SELECT student
FROM Grade AS Gr1
WHERE NOT EXISTS ( SELECT score
FROM Grade AS Gr2
WHERE Gr1.student = Gr2.student AND
NOT (score = 'AA' OR score = 'AAA'))
```

Different database schemes can correspond to the same domain model, and even the same database scheme can be translated to different logical vocabularies. The chosen scheme has effect on the simplicity of the SQL queries, and in the same way the logical vocabulary has effect on the simplicity of modelling or execution speed of the inferences. We propose two possible ways to convert a scheme into a vocabulary.

- The “classical” way: we choose an *n*-ary predicate symbol per table with *n* columns.
- The “object oriented” way: we choose the table name *table* as a unary type-predicate and the column names as *field_i* as binary relations. For each row of the table a new “identifier” is created who represent the objects.

The SQL-table *Grade(student, subject, grade)* can be represented as a ternary relation as in the student database. The alternative is to view this table as a set of relations in Figure 2 for the predicate symbols *Grade/1*, *G2Student/2*, *G2Course/2* and *G2Score/2*.

Exercise 4 (continued). Translate the previous SQL query to this new vocabulary.

Exercise 5. Take a table for representing employees in a company: *Employee(id, surname, name, street, houseNb, city, zip, birthDate, job, salary, nbOfServedYears)*. This is a relation with 11 arguments. Suppose we want to know which employees earn more than 100.000 a year. In SQL this can be written down as:

```
SELECT id
FROM Employee
WHERE Employee.salary ≥ 100.000;
```

Grade			⇒	Grade ^I		G2Student ^I		G2Course ^I		G2Score ^I	
Sam	CS148	AAA		G1	G1	Sam	G1	CS148	G1	AAA	
Bill	CS148	D		G2	G2	Bill	G2	CS148	G2	D	
Jill	CS148	A		G3	G3	Jill	G3	CS148	G3	A	
Jack	CS148	C		G4	G4	Jack	G4	CS148	G4	C	
Flo	CS230	AA		G5	G5	Flo	G5	CS230	G5	AA	
May	CS230	AA		G6	G6	May	G6	CS230	G6	AA	
Bill	CS230	F		G7	G7	Bill	G7	CS230	G7	F	
Ann	CS230	C		G8	G8	Ann	G8	CS230	G8	C	
Jill	M100	B		G9	G9	Jill	G9	M100	G9	B	
Sam	M100	AA		G10	G10	Sam	G10	M100	G10	AA	
Flo	M100	D		G11	G11	Flo	G11	M100	G11	D	
Flo	M100	B	G12	G12	Flo	G12	M100	G12	B		

Figure 2: A database as an alternative structure

Translate this query to logic, once with the classical vocabulary, and once with the alternative vocabulary. Which of those lies closer to the syntax of the SQL query?

Exercise 6. Suppose the table of Employees has a primary key, consisting of the first column. Which improvements can be made to the logical vocabulary and structure? (Hint: which property do the alternative vocabulary and a primary key have in common?)

[EXTRA] Exercise 7. Suppose a database with two tables:

- *suppliers(supplier_id, supplier_name)*
- *orders(supplier_id, quantity, price)*

The column *supplier_id* is a primary key in the first table.

Convert this database to a logical vocabulary and structure. Using this vocabulary, convert this query to logic.

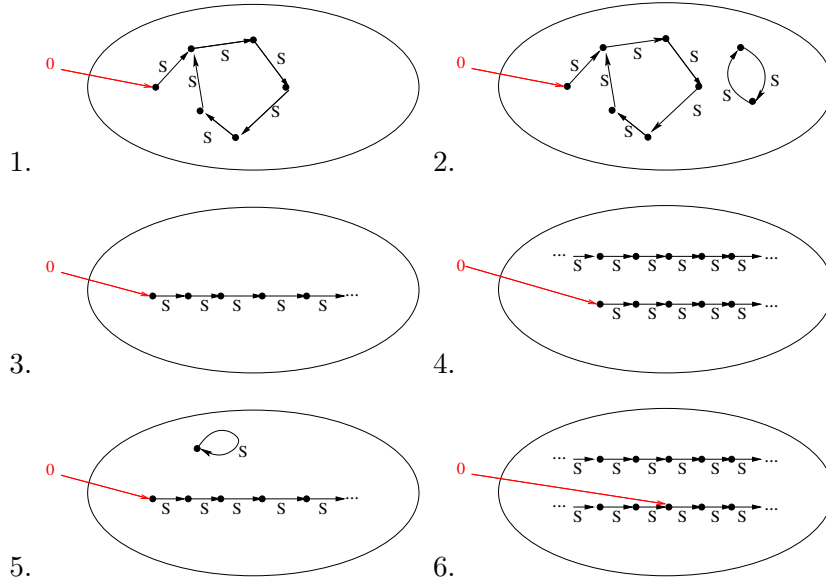
```

SELECT orders.quantity, orders.price
FROM orders
WHERE EXISTS(SELECT suppliers.supplier_id
FROM suppliers
WHERE suppliers.supplier_name = 'IBM'
and suppliers.supplier_id = orders.supplier_id);

```

3 UNA and DCA

Exercise 8. Here are 6 images of $\{0, S\}$ -structures. In which of these images is the domain closure axiom satisfied? In which of these is the unique names axiom satisfied?



Exercise 9. Define $\text{UNA}(\tau)$ and $\text{DCA}(\tau)$ for $\tau = \{ \text{Nil} : \text{list}, \text{Cons} : A \times \text{list} \rightarrow \text{list} \}$. Here *list* is the type of lists over *A*. This axiom expresses that different lists are different.

Exercise 10. Define the UNA and DCA in IDP for a type consisting of the four wind directions (North, West, South, East). This means you can not use constructed types.