Databases from a logical perspective

18 oktober 2016

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

Ray	CS230
Hec	CS230
Sue	M100
Sue	M200
Pat	CS238

Prerequ[a]

CS230	CS238
CS148	CS230
M100	M200

[a]Prerequ(a,b) means a is a prerequisite for b

Enrolled

Jill	CS230
Jack	CS230
Sam	CS230
Bill	CS230
May	CS238
Ann	CS238
Tom	M100
Ann	M100
Jill	M200
Sam	M200
Flo	M200

Grade

Sam	CS148	AAA
Bill	CS148	D
Jill	CS148	A
Jack	CS148	C
Flo	CS230	AA
May	CS230	AA
Bill	CS230	F
Ann	CS230	C
Jill	M100	B
Sam	M100	AA
Flo	M100	D
Flo	M100	B

PassGrade

AAA
AA
A
B
C

Figuur 1: The student database

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

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

The domain is

$$Domein_I = \{Ray, Hec, \ldots, CS230, \ldots, AAA, \ldots, Jill, \ldots\}$$

The interpretation of I consists of

- The interpretation of all constants: $Ray^I = Ray, \ldots, CS230^I = CS230, \ldots, AAA^I = AAA, \ldots$
- The relation of tuples of domain elements $Instructor^I, Grade^I, \ldots$ 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?*
   Solution:
   $$(\exists x) Prerequ(x, CS238)$$
   Answer: True.

2. *Has May passed for CS230 ?* **Solution:**
   $$(\exists x)(Grade(May, CS230, x) \wedge PassingGrade(x))$$
   Answer: True.

3. *Are all the courses which are (direct) prequisite for M100 instructed by Ray?*
   Solution:
   $$(\forall x)(Prerequ(x, M100) \Rightarrow Instructor(Ray, x))$$
   Strange question! $M100$ has no prerequisites!
   Answer: Trivially true. He instructs each of the 0 courses.

4. *Has everyone that is enrolled in CS230 passed for at least one course instructed by Sue?*
   Solution:
   $$(\forall x)(Enrolled(x, CS230) \Rightarrow (\exists w)(\quad Instructor(Sue, w) \wedge (\exists s)(Grade(x, w, s) \wedge PassingGrade(s))))$$
   Answer: False!

5. *Did John do an exam for CS230?*
   Solution:
   $$(\exists s)Grade(John, CS230, s)$$
   Note: John is not an element of the domain of the database, a database will answer: false, but in logic the truth value of this sentence will be undefined. Note: in the IDP system you cannot write a query like this (since John is not in the vocabulary either)

6. *Are there students taking a course from every instructor?*
   Solution:

   $$(\exists x)(\forall y)((\exists w)Instructor(y, w) \Rightarrow (\exists w)(Instructor(y, w) \wedge Enrolled(x, w)))$$

   Answer: False

7. *Is there are student who is enrolled in exactly 1 course?*
   Solution:
   $$(\exists x)(\exists 1 w)Enrolled(x, w)$$
   Answer: True

Exercise 2 An *n-ary* query is a symbolic definition of a relation of the form $\{(x_1, \ldots, x_n) : A\}$. 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.*
   Solution:

   $$\{(x, y) : (\exists z)(Grade(x, y, z) \wedge PassingGrade(z))\}$$

   A database will answer this question with the following relation:

Sam	CS148
Jill	CS148
Jack	CS148
Flo	CS230
May	CS230
Ann	CS230
Jill	M100
Sam	M100
Flo	M100

2. *All students who follow exactly one course*
   Solution:
   $$\{x : (\exists 1 w)(Enrolled(x, w)\}$$

3. *All students who are enrolled in CS230 and have passed for all the courses they did up til now.*
   Solution:

   $\{x : Enrolled(x, CS230) \wedge (\forall y)((\exists z)(Grade(x, y, z)) \Rightarrow (\exists z)(Grade(x, y, z) \wedge PassingGrade(z)))$

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*
   Solution:

   $$\neg(\exists w) Prerequ(w, w)$$

   Satisfied.

2. *Nobody can have more than one grade for a course.*
   Solution:

   $$\neg(\exists x)(\exists w)(\exists s)(\exists t)(Grade(x, w, s) \wedge Grade(x, w, t) \wedge s \neq t)$$

   Not satisfied

3. No student can be an instructor. (A student is someone who is enrolled in a course)
   Solution:

   $$\neg(\exists x)((\exists w) Instructor(x, w) \wedge (\exists w) Enrolled(x, w))$$

   Satisfied In this sentence $(\exists w) Instructor(x, w)$ expresses that x is an instructor and $(\exists w) Enrolled(x, w))$ expresses that x is a student. Note that the w in those subsentences is unrelated. We have similar integrity constraints to ensure that subjects, students, instructors and grades are pairwise disjunct, so they contain no common elements.

4. *Everyone who is enrolled for a course needs to have passed all courses which are (direct) needed prerequisites for that course.*
   Solution:

   $(\forall x)(\forall w)(Enrolled(x, w) \Rightarrow (\forall t)(Prerequ(t, w) \Rightarrow (\exists s)(Grade(x, t, s) \wedge PassGrade(s))))$

   Not valid.

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:

$$table(field_1, field_2, \ldots, veld_n)$$

Here *table* is the name of the table and $field_i$ is the name of the ith column of the table. E.g. $Grade(student, vak, score)$, $Enrolled(student, score)$.

Exercise 5. 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'))

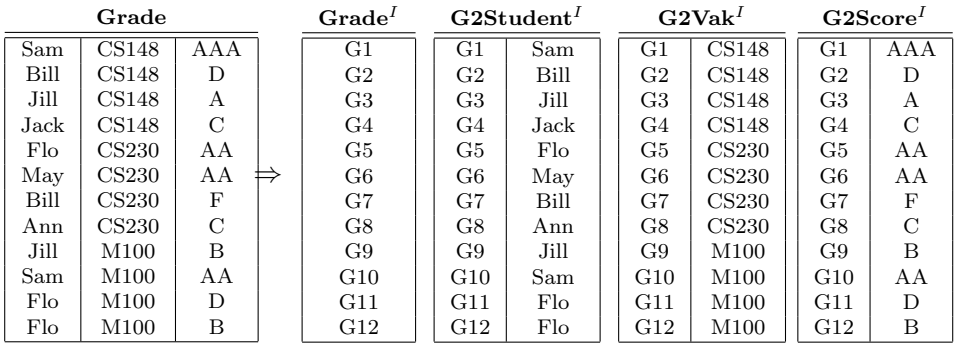
Solution:

$$\{x : (\exists y)(\exists z)Grade(x, y, z) \wedge (\forall y)(\forall z)[Grade(x, y, z) \Rightarrow z = AA \vee z = AAA]\}$$

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 $Grade/1$, $G2Student/2$, $G2Vak/2$ and $G2Score/2$.



Grade		
Sam	CS148	AAA
Bill	CS148	D
Jill	CS148	A
Jack	CS148	C
Flo	CS230	AA
May	CS230	AA
Bill	CS230	F
Ann	CS230	C
Jill	M100	B
Sam	M100	AA
Flo	M100	D
Flo	M100	B

⇒

$\mathbf{Grade}^I$
G1
G2
G3
G4
G5
G6
G7
G8
G9
G10
G11
G12

$\mathbf{G2Student}^I$	
G1	Sam
G2	Bill
G3	Jill
G4	Jack
G5	Flo
G6	May
G7	Bill
G8	Ann
G9	Jill
G10	Sam
G11	Flo
G12	Flo

$\mathbf{G2Vak}^I$	
G1	CS148
G2	CS148
G3	CS148
G4	CS148
G5	CS230
G6	CS230
G7	CS230
G8	CS230
G9	M100
G10	M100
G11	M100
G12	M100

$\mathbf{G2Score}^I$	
G1	AAA
G2	D
G3	A
G4	C
G5	AA
G6	AA
G7	F
G8	C
G9	B
G10	AA
G11	D
G12	B

Figuur 2: A database as an alternative structure

Exercise 5 (continued). Translate the previous SQL query to this new vocabulary.

Exercise 6. 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:

$$SELECT\ id$$
$$FROM\ Werknemer$$
$$WHERE\ Werknemer.loon \geq 100.000;$$

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?

Solution:

$$\{id : (\exists vn)(\exists n)(\exists st)(\exists stnr)(\exists g)(\exists z)(\exists geb)(\exists j)(\exists l)(\exists a) \\ (Werknemer(id, vn, n, st, stnr, g, z, geb, j, l, a) \wedge l > 100.000)\}$$

$$\{id : (\exists w)[Werknemerid(w, id) \wedge (\exists l)(Werknemerloon(w, l) \wedge l \geq 100.000)]\}$$

Exercise 7. 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 8. Suppose a database with two tables $suppliers(supplier\_id, supplier\_name)$ and $orders(supplier\_id, quantity, price)$. 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.

$$
\begin{array}{l}
SELECT\ orders.quantity, orders.price \\
FROM\ orders \\
WHERE\ EXISTS(SELECT\ suppliers.supplier\_id \\
FROM\ suppliers \\
WHERE\ suppliers.supplier\_name =' IBM' \\
and\ suppliers.supplier\_id = orders.supplier\_id);
\end{array}
$$

3 A database as a logical theory

Aside from viewing a database as a structure, we can also take the viewpoint of seeing them as a logical theory, independent of a structure. This allows to do (for example) theorem proving.

A translation of a database to a theory is successful if a theorem prover can answer all our queries with the same answer as a database system.

What is the purpose of this? Do we mean that database systems should be replaced with theorem provers? That would be ridiculous. No, we rather mean that a database system can -also- be regarded as a super efficient theorem prover for a very specific and simple kind of information, the kind of information we find inside of databases. We simply want to understand which kinds of information can be represented in a database and which cannot. We will notice that there are a few important and unexpected lessons to learn about the modelling of information in logic.

Exercise 9. Construct a theory T_{DB} which expresses exactly the information of the student database. Do not use constructed types, use constants to represent the students, courses, grades and teachers.

Exercise 10. The following queries are true in the database, are they also true in the theory you just constructed? Prove that they are true or give a structure in which they aren't. (You don't need to improve your theory if some queries fail, this will be assessed later in this assignment)

- $\neg Ray = Hec$
- $\neg Instructor(Ray, CS238)$
- $\neg(\exists x) Prerequ(x, x)$ - an integrity constraint
- $(\forall x)(Prerequ(x, CS238) \Rightarrow Instructor(Ray, x))$

Solution:

Proof. For each query in this list we need to find a model I of T_{DB} which does not satisfy this query. For the first queries we choose I as follows:

- $D_I = \{a\}$; in other words, there is only one domain element.
- $Ray^I = Hec^I = \cdots = CS230^I = \cdots = AAA^I = \cdots = Sam^I = \cdots = a$: a is the interpretation of every constant.
- $PassingGrade^I = \{a\}$.
- $Instructor^I = Prerequ^I = Enrolled^I = \{(a, a)\}$.
- $Grade^I = \{(a, a, a)\}$.

It is easy to see that this structure is a model of T_{DB}. The follows from the observation that really any atomic formula over Σ and every equality atom $c = c'$ is true in I, so also the formulas of T_{DB}. For the first three queries, it holds that they are not true in I. E.g. $I \not\models \neg(\exists x)Prerequ(x, x)$ because $I \models Prerequ(a, a)$.

The query $(\forall x)(Prerequ(x, CS238) \Rightarrow Instructor(Ray, x))$ is satisfied in I because for 'each' domain element (i.e. a) this conclusion holds. So we have to find another structure in which T_{DB} holds but in which this formula does not hold.

Take the structure I' which results from adding b from I to the domain and adding one tuple to $Prerequ^I$:

$$Prerequ^{I'} = \{(a, a), (b, a)\}$$

I' still satisfies T_{DB} and $I' \not\models Prerequ(b, CS238) \Rightarrow Instructor(Ray, b)$ holds, so the fourth query is not satisfied. Q.E.D.

If we look to the queries of the student database, we can prove that no query that contains $\Rightarrow, \neq, \Leftrightarrow$ or $\forall$ is provable from T_{DB}. Only the most simple true queries can be answered with a theorem prover starting from T_{DB}.

Here we bump again into an important aspect of modelling: *implicit knowledge*. There are three kinds of implicit information in the database system in a database system which needs to be modelled explicitly in a logical theory (because logical theories give the user more freedom).

- In a database different strings point to different objects. So we know that $Ray, Hec, Sam, CS230, AAA, \ldots$ are all different things. This was not expressed in T_{DB}, so the theorem prover couldn't know that $\neg Ray = Hec$. These inequalities are called the *Unique Name Axioms.*

  Exercise 11. Express this information in logic

- We have not expressed that there are no other objects in the universe that are applicable, than the ones in our database domain:

  $$\{Ray, \ldots, Sam, \ldots, CS230, \ldots, AAA, \ldots\}.$$

  The axiom to express this, is called the *Domain Closure Axiom.*

Exercise 12. Express this information in logic

- Our logical theory contained the information which tuples were in the tables but not which tuples were not in the tables. For example: the theory does not express that $\neg Prerequ(CS230, CS230)$ so the theorem prover can not prove that no course is a prerequisite of itself. There are more tuples which are not in the table than those who are, and over those no information is present in our theory. To express exactly which tuples are present and which aren't, we need *definitions*.

Exercise 13. Express this information in logic

Solution:

Definition 3.1. Let S be a set of constants. The Unique Name Axioms for S, notated by $UNA(S)$, is the set of all formulas $\neg C = C'$ for each pair of different constants $C, C' \in S$.

Proposition 3.1. *In each model I of $UNA(S)$ it holds for each pair of constant $C \neq C' \in S$ that $C^I \neq C'^I$. A database structure I satisfies to $UNA(S)$ met $S = \{c' | c \in D_I\}$*

Proof. Trivial.

Definition 3.2. Let $S = \{C_1, \ldots, C_n\}$ be a non-empty set of constant. The Domain Closure Axiom for S, notated by $DCA(S)$, is the sentence:

$$(\forall x)(x = C_1 \vee \cdots \vee x = C_n)$$

Proposition 3.2. *In each model I satisfying $DCA(S)$ there exists a constant C for every domain element $d \in D_I$ such that $C^I = d$. Also if the vocabulary Σ_I of I contains other constants and function symbols, it holds that for each term t that a constant $C \in S$ exists such that $t^I = C^I$ holds. A database structure I satisfies to $DCA(S)$ with $S = \{c' | c \in D_I\}$.*

Proof. Trivial from the $\forall$-rule and the $\vee$-rule from the definition of truth.

Proposition 3.3. *A structure I is a model of $UNA(S) \wedge DCA(S)$ iff it contains a domain with n elements such that for every domain element $d \in D_I$ there exists a unique constant $C \in S$ such that $C^I = d$. For each pair of models I, J there exists a bijection $b : D_I \to D_J$ such that for each $d \in D_I$ it holds that if $C^I = d$ then $C^J = b(d)$.*

Proof. Through the combination of the previous two propositions.

The following definition is in the context of a database-structure I of the vocabulary Σ. We know that for each object c in the domain of the I-database a constant $c \in \Sigma$ exists.

Definition 3.3. Let R be the name of an n-ary table consisting of a set of m-tuples from the database domain $(c_{1,1}, \ldots, c_{1,n}), \ldots, (c_{m,1}, \ldots, c_{m,n})$. The *definition* of R, notated by $Def(R)$, is the logical sentence:

$$\begin{aligned}(\forall x_1)\ldots(\forall x_n)(R(x_1,\ldots,x_n) \Leftrightarrow\ & (x_1 = c_{1,1} \wedge \cdots \wedge x_n = c_{1,n}) \\ & \vee \cdots \vee \\ & (x_1 = c_{m,1} \wedge \cdots \wedge x_n = c_{m,n})\end{aligned}$$

Example 3.1. For example, $Def(Prerequ)$ is:

$$\begin{aligned}(\forall x)(\forall y)[Prerequ(x,y) \Leftrightarrow\ & (x = CS148 \wedge y = CS230])\vee \\ & (x = CS230 \wedge y = CS238)\vee \\ & (x = M100 \wedge y = M200)]\end{aligned}$$

The next proposition is not about a database structure I but about all kinds of structures which are a model of $Def(R)$.

Proposition 3.4. *A structure J is a model of $Def(R)$ if it holds that for each tuple $(d_1, \ldots, d_n) \in R^J$ a tuple of constants $(c_1, \ldots, c_n) \in R^I$ exists such that ${c_1}^J = d_1, \ldots, {c_n}^J = d_n$. A database structure I is a model of $Def(R)$.*

Proof. Directly from the application of the $\forall$- and $\Leftrightarrow$-rule on $Def(R)$.

Definition 3.4. Let I be a database with vocabulary Σ and constants S. So, I is a structure with a finite domain such that Σ consists of all table names and for each domain element $d \in D_I$, a constant d such that $d^I = d$. Define the theory of I, notated by $Theo(I)$, as the set of logical sentences:

- $UNA(S)$;
- $DCA(S)$;
- $Def(P^I)$, for each predicate symbol $P \in \Sigma$.

$Theo(I)$ is another way to model the information of a database in logic. We will prove that this theory is correct. More specifically, this theory has essentially only one model, namely I itself.

This is not completely right, this theory has many models, but the all have the same inner structure as I. We say that all models are *isomorphic* to I.

Definition 3.5. Two structures I, J of the same vocabulary are **isomorphic** iff there exist a bijection b from D_I to D_J which preserves the interpretation of every symbol:

- if $C^I = d$ then $C^J = b(d)$
- if $F^I(d_1, \ldots, d_n) = d$ then $F^J(b(d_1), \ldots, b(d_n)) = b(d)$, for each function F;
- if $I \models P(d_1, \ldots, d_n)$ then $J \models P(b(d_1), \ldots, b(d_n))$ holds for each predicate P.

Property 3.1. In isomorphic structures the same formulas are true.

We will not prove this but the proof is not difficult. It is a proof by induction completely analog to the the proof of the Generalising property.

Oefening 3.1. *Prove this proposition.*

Theorem 3.1. *Given a database structure I with vocabulary Σ.*
(a) A structure J of Σ is a model of $Theo(I)$ iff J and I are isomorphic.
(b) For each logical sentence A of Σ it holds that $Theo(I) \models A$ iff $I \models A$.

In words: A is logical consequence of the theory $Theo(I)$ iff A is true in structure I. This means that -in principle- a theorem prover from $Theo(I)$ can answer the same queries as a database system of ı. This means that we can view a database system as a super efficient theorem prover for the theory $Theo(I)$.

Proof.

(a) From Proposition 3.3 it follows that J is a model of $UNA(S) \wedge DCA(S)$ iff there exists a bijection between the domain of J and I such that $b(c^J) = c = c^I$. From Proposition 3.4 it follows that such a J satisfies the definition $Def(R)$ iff the image b applied to the tuples of R^J are exactly the tuples of R^I.

(b) Because the same formulas are true in all isomorphic structures, (b) follows out of (a). Q.E.D.

This proposition guarantees us that $Theo(I)$ contains all information in database I.

This information is *logically consistent* and *complete*. The information is consistent because there is a model, and the information is complete because the model is unique (modulo isomorphism) and so for each sentence A of Σ it holds that either A, or $\neg A$ is a logical consequence of $Theo(DB)$. This follows directly from Proposition 3.1 and the fact that $I \models A$ or $I \models \neg A$.

3.1 Null-values.

Suppose that in a table of the database, one or more values are lacking. In classical databases, one places NULL on those positions. One calls this a null-value. This means that the value for a position is unknown, for example:

Instructor

Ray	NULL
Hec	CS230
Sue	NULL
NULL	M200
Pat	CS238

Intuitively this means the following: the table represents all information about who instructs which courses except we don't know which course is instructed by Ray and we don't know who instructs M200. Note that it is possible that Ray instructs M200.

Exercise 14. Modify the logical theory so it contains all information from the $Instructor$ table.

Solution: The basic idea is the following: we propose to introduce a new constant n_i for each occurrence of NULL. So we can split the constants of Σ in the set S of original constants of which we know the identity, and the new null-constant $n_1, \ldots, n_l$. The theory is modified as follows:

- $UNA(S)$: this axiom does not change but is only applicable to constants with a known identity; the theory excludes the possibility that $Ray = Hec$ but not that $Ray = n_2$.
- $DCA(S)$: this implies that n_i has to be equal to one of the constants from S with known identity.
- $Def(P)$, for each predicate $P \in \Sigma$: these formulas don't change either. They imply that Ray instructs the course n_1, and that somebody n_2

instructs the course $M200$. This formula is the following:

$$\begin{aligned}(\forall x)(\forall y)(Instructor(x,y) \Leftrightarrow\ & (x = Ray \wedge y = n_1) \vee \\ & (x = Hec \wedge y = CS230) \vee \\ & (x = Sue \wedge y = M100) \vee \\ & (x = n_2 \wedge y = M200) \vee \\ & (x = Pat \wedge y = CS238)\end{aligned}$$

Exercise 15. Suppose we want the additional constraint: the unknown courses that Ray and Sue instruct are different. Express this in your solution.

Exercise 16. Suppose we want the additional constraint: The unknown course given by Ray is either CS230 or CS 238. Express this in your solution.