Universidade Nova de Lisboa
OMNIS CIVITAS CONTRA SE DIVISA NON STABIT
Faculdade de Ciências e Tecnologia

Functional Dependences; Normal Forms – Summary and exercises

(13 Jan 2017. Aula prática 2)

The data model we have designed so far has not been and should not be determined by the particular possible applications related to the video club: particular queries, particular reports, etc. It should only be determined by Normalization based on the set of explicit and implicit (deductible) functional dependencies. What we have done so far has been the indirect application of this Normalization by resorting to some practical rules, but not less robust. Let us now verify that we have done it correctly, using Normalization.

The so-called Normalization serves to avoid redundancies in the Data Model, causing inconsistencies, and to facilitate access to data in a rigorous and complete way. Here are the most important Normal Forms that constitute it.

1st Normal Form

An R schema (or, if you like, an R table) is in the First Normal Form if the domains of its attributes are atomic, that is, if their elements are indivisible units. For example, suppose that the member table had an attribute called filhos_sócio (member's children) where the names of the children were written for each member: John, Mary, Louis for sócio whose number (num_sócio) is equal to 5, and Pedro Manuel, Joan for sócio with num_sócio equal to 6. Clearly, these contents are not atomic because it is necessary to remove from each chain the individual names of the children. This table would not be in the 1st Normal Form. Another example of failure of the 1st Normal Form is related to the zip code. Thus, let us suppose that the sócio table had a separate field called cod_postal_sócio which was intended to contain the zip code of the sócio's address and that it was systematically filled with the number-location structure. Here too the content is not atomic. This fault is not serious but generates redundancy, since if there are at least two sócios with the same zip code, the location will appear repeated

2ª Normal Form

An R schema is in the 2nd Normal Form if and only if it is in the 1st and if each non-key R attribute functionally depends on the whole key. In practice this requirement is satisfied if R is in the 3rd Normal Form, as we shall understand.

Armstrong Rules

When we are given a set of functional dependencies, there are usually other dependencies that can be concluded from this set and from coherent rules. Armstrong's rules are coherent and their exhaustive application to a set of functional dependencies, form what is called the closing (complete set) of functional dependencies that can be deduced in a coherent way. Here are the rules:

If $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ (reflexivity)

If $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$ (augmentation)

If $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (transitivity)

3ª Normal Form

A schema R is in the 3rd Normal Form if and only if for all the functional dependence $\alpha \rightarrow \beta \in$

F^+, at least one of the following conditions is true (F^+ is the closing (full set) of all functional dependencies, the direct ones and deductible by Armstrong rules.):

. $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)

. α is superkey of R (i.e.,$\alpha \rightarrow R \in F^+$)

. every attribute $A \subseteq (\beta - \alpha)$ is contained in a candidate key of R.

(NOTE: Each of these attributes may belong to a different candidate key)

In other words, R is in the 3rd Normal Form if none of its attributes depends on other attributes that are not part of one of the candidate keys of R.

To illustrate the 3rd Normal Form, let's assume that the functional dependency *ano_filme* → *preço_dia_filme* ∈ F+: in this case, this dependency would place the filme table outside the 3rd normal form, since the preço_dia_filme attribute would depend on an attribute (ano_filme) that is not part of any candidate key of *filme*.

We can consider that the 3rd Normal Form is the minimum goal to be achieved in the Normalization Process of a Data Model. The 3rd Normal Form does not avoid redundancies.

Boyce-Codd Normal Form (BCNF)

A schema R is said in BCNF, for a set of dependencies F, if and only if for all dependences on F^+ of the form α → β, where α ⊆ R and β ⊆ R, at least one of the conditions is true:

. α → β is trivial (i.e., β⊆α)

. α is a superkey of R (i.e., α → R ∈ F +)

In other words, there can only be nontrivial functional dependencies applicable to an R scheme if the left part of the dependency is a superkey. This requirement is stronger than that determined by the 3rd Normal Form. If a R scheme is in BCNF it is also in the 3rd Normal Form. The BCNF avoids redundancies and is a goal to be achieved in the design of Data Models.

Let's say we had included the nome_actor attribute in the filme_actor table. Now there would be more new functional dependencies to take into account, for example: *cod_filme,nome_actor → cod_filme,nome_actor,cod_actor*, thus being another candidate key, and *cod_actor → nome_actor*, which implies that the filme_actor table would be in the 3rd Normal Form (*nome_actor* em *cod_actor → nome_actor* would be part of a candidate key) but would not be in BCNF precisely because *cod_actor* in *cod_actor → nome_actor* is not superkey in filme_actor.

Exercise 1

Based on functional dependency analysis, create a Data Model in BCNF for a physician's office. Consider consultations, patients, doctors and specialties (stomatology, pneumology, etc.). Every doctor can be a specialist in more than one specialty. During appointments, doctors prescribe drugs (one or more). There are also appointment bookings. Consider also as valid the following functional dependencies of the set F:

cod_doctor → name_doctor
(cod_doctor, name_doctor) → (cod_speciality, name_speciality)
name_speciality → cod_speciality
cod_speciality → name_speciality
cod_drug → name_drug
cod_patient → name_patient

(date_appointment, time_appointment cod_doctor) → (cod_patient, name_patient, date_booking cod_speciality)
(date_appointment, time_appointment cod_patient) → (cod_doctor, name_doctor, date_booking, cod_speciality)
(date_appointment, time_appointment, cod_doctor, cod_drug) → qt_prescription

First of all, it is convenient to find a canonical cover G of F.

Thus, we see that *name_doctor* is redundant to the left in *(cod_doctor, name_doctor) → (cod_speciality, name_speciality),* since *F* logically implies a coverage in which the DF containing the redundancy is replaced by another one without this redundancy (see slide 7.23) , ie: by pseudo-transitivity (PT) the DFs *cod_doctor → name_doctor* and *(cod_doctor, name_doctor) → (cod_speciality, name_speciality)* derive

cod_doctor → (cod_speciality, name_speciality). Schematically,

$$\frac{\text{cod\_doctor} \rightarrow \text{name\_doctor} \qquad \text{(cod\_doctor, name\_doctor)} \rightarrow \text{(cod\_speciality, name\_speciality)}}{\text{cod\_doctor} \rightarrow \text{(cod\_speciality, name\_speciality)}}\ (PT)$$

On the other hand, in this new DF, there is still a redundancy on the left: *name_speciality.*

In fact, if we substitute this DF for another without this redundancy, we will obtain a new coverage that logically implies the previous one. Echematically:

$$\frac{\text{cod\_doctor} \rightarrow \text{cod\_speciality} \qquad \text{cod\_speciality} \rightarrow \text{name\_speciality}}{\text{cod\_doctor} \rightarrow \text{name\_speciality}}\ (T)$$

$$\frac{\text{cod\_doctor} \rightarrow \text{name\_speciality} \qquad \text{cod\_doctor} \rightarrow \text{cod\_speciality}}{\text{cod\_doctor} \rightarrow \text{(cod\_speciality, name\_speciality)}}\ (U)$$

Applying the same mechanisms of elimination of redundancy to the left and to the right, together with the union of DFs with the same left part, canonical coverage G:

cod_doctor → (name_doctor, cod_speciality)
cod_speciality →name_speciality
name_speciality → cod_speciality
cod_drug → name_drug
cod_patient → Name_patient
(date_appointment, time_appointment, cod_doctor) → (cod_patient, date_booking)
(date_appointment, time_appointment, cod_patient) → (cod_doctor, date_booking)
(date_appointment, time_appointment, cod_doctor, cod_drug) → qt_prescription

The initial schema is *R=(cod_doctor, name_doctor, cod_speciality, name_speciality, cod_drug, name_drug, cod_patient, name_patient, date_appointment, time_appointment, date_booking, qt_prescription).* One of the keys of *R* is (*date_appointment, time_appointment, cod_doctor, cod_drug)* as {*date_appointment, time_appointment, cod_doctor, cod_drug}+ = R.* Another key (a set of attributes whose closure includes all the attributes of *R*) is *(date_appointment, time_appointment, cod_patient, cod_drug).* It turns out that *R* is not in the BCNF since, for example, the functional dependency (DF) cod_patient → name_patient violates this normal form because *cod_patient* is not a key in *R* nor this DF is trivial. Thus, it is necessary to decompose schema *R*.

Decomposition (slide slide 7.36):
Result:=R
*R =R*1
α *= cod_patient* β*= name_patient*
*Result:= (Result –R*1*)* ∪ *(R*1*- β)* ∪ *(α, β)*
Thus, *R*2*=(cod_patient, name_patient*), *R*3=(*cod_doctor, name_doctor, cod_speciality, name_speciality, cod_drug, name_drug, cod_patient, date_appointment, time_appointment, date_booking, qt_prescription).* R2 is in BCNF because no other DF applies to R2.

However, R3 is not in the BCNF for violation, for example, of *cod_speciality → name_speciality*. This makes the decomposition evolve to: R2 ∪ R4 ∪ R5, where R4 = *(cod_speciality, name_speciality)* and R5 = *(cod_doctor, name_doctor, cod_speciality, cod_drug, name_drug, cod_patient, date_appointment, date_book, date_booking, qt_prescription)*. R4 is in the BCNF but R5 is not because *cod_doctor → (name_doctor, cod_speciality)* violates the BCNF since *cod_doctor* is not key in R5 nor the DF in question is trivial. In the next iteration of the algorithm, Result = R2 ∪ R4 ∪ R6 ∪ R7, with R6 = *(cod_doctor, name_doctor, cod_speciality)* that is in BCNF, and R7 = *(cod_doctor, cod_drug, cod_drug, name_drug, cod_patient,*

date_appointment, appointment_time, date_booking, qt_prescription). However, *cod_drug* → *name_drug* prevents R7 from being in the BCNF. In the next iteration, Result = R2 ∪ R4 ∪ R6 ∪ R8 ∪ R9, with R8 = *(cod_drug, name_drug),* which is in BCNF, and R9 = *(doc_doctor, cod_drug, cod_patient, date_appointment, date_book, date_booking, qt_prescription).* But R9 is not in BCNF because in *(date_appointment, aptation_time, cod_doctor)* → *(cod_patient, date_booking, date_appointment, aptation_time, cod_doctor)* the left part of the DF is not a key in R9 nor is DF trivial. So Result = R2 ∪ R4 ∪ R6 ∪ R8 ∪ R10 ∪ R11 with R10 = *(date_appointment, aptation_time, cod_doctor, cod_patient, date_booking)* that is in BCNF and R11 = *(doc_doctor, date_appointment, time_ appointment, cod_drug, qt_prescription)*, which is in BCNF. Thus, the database will consist of the following schemas that, given its composition, can have the following names of tables and contents:

Patient =(*cod_patient, name_patient*)
Speciality =*(cod_speciality, name_speciality*)
Doctor=(*cod_doctor, name_doctor, cod_speciality*)
Drug=*(cod_drug, name_drug)*
Appointment=(date_appointment, time_appointment, cod_doctor, cod_patient, date_booking)
Prescription=(cod_doctor, date_appointment, time_appointment, cod_drug , qt_prescription)

Quanto à preservação das dependências, é fácil ver que o fecho da união das DFs que se aplicam a cada esquema coincide com com o fecho da cobertura G, isto é:

Regarding the preservation of the dependencies, it is easy to see that the closure of the union of the DFs that apply to each schema coincides with the closure of the cover G, that is:

F_Patient={ *cod_doctor → name_doctor, cod_speciality*};
F_speciality=*{cod_speciality→name_speciality, name_speciality→ cod_speciality}*
F_Doctor={ cod_doctor→ name_doctor, cod_speciality}
F_Drug={ cod_drug → name_drug}
F_Booking={date_appointment, time_appointment, cod_doctor → cod_patient, date_booking,
date_appointment, time_appointment, cod_patient → cod_doctor, date_booking}
F_Prescription={ date_appointment, time_appointment, cod_doctor, cod_drug → qt_prescription}

In other words, {F_Patient ∪ F_speciality ∪ *F_Doctor* ∪ *F_Drug* ∪ *F_Booking* ∪ *F_Prescription}+ = G+ .*

Exercise 2:

a) Make sure the video club database, drawn through the E-R Model, is in the BCNF.
Consider the following DFs of *F* as valid:

cod_género → nome_género
cod_editora → nome_editora
cod_actor → nome_actor
cod_realizador → nome_realizador
cod_filme→ (nome_filme, ano_filme, preço_dia_filme, dias_sem_multa_filme, multa_dia_filme, cod_género, cod_editora)
(num_copia, cod_filme,_data_aluguer) → (num_sócio, data_devolução, estado_devolução)
num_sócio → (nome_sócio, bi_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio)
bi_sócio → (nome_sócio, num_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio)

Here are the tables:

Sócio=(num_sócio, nome_sócio, bi_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio)
Género=(cod_género, nome_género)
Editora=(cod_editora, nome_editora)
Actor=(cod_actor, nome_actor)
Realizador=(cod_realizador, nome_realizador)
Filme=(cod_filme, nome_filme, ano_filme, preço_dia_filme, dias_sem_multa_filme, multa_dia_filme, cod_género, cod_editora)
Filme_actor=(cod_filme,cod_actor)Ffilme_realizador=(cod_filme,cod_realizador)
Cópia=(cod_filme,num_cópia)
Aluguer=(cod_filme,num_cópia,data_aluguer,num_sócio)
Devolução=(cod_filme, num_cópia, data_aluguer, data_devolução, estado_devolução)

We will now have to check if each schema (corresponding to each table) is in the BCNF. Thus, following slide 7.38, we see that it is not necessary to compute F+ (the complete closure of F) but it is necessary to compute the closure of each subset of the attributes of each scheme *R* and see if that closure is limited to itself or to the set *R* If this does not happen, *R* is not in BCNF.

Thus, in the género scheme, we have the cod_género attribute, whose closure {*cod_género}+* = {cod_género, nome_genero} since it contains cod_género by reflexivity and, by application of DF cod_género → nome_genero, it contains nome_género, ie, the closure contains the whole

schema. On the other hand, the closure of nome_género is just {nome_género }. No other DF applies to the schema, so Gender is in the BCNF.
An analogous analysis proves that the Editora, Actor e Realizador schemas are also in BCNF. As for the sócio schema, only the closure of *num_sócio*, the closure of *bi_sócio*, and the closure of the subsets containing one or two of these attributes do not only contain the attributes themselves (for example, {morada_sócio, tlf_sócio}+ = { morada_sócio, tlf_sócio}). As for {num_sócio}+ it equals to {num_sócio, nome_sócio, bi_sócio, data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio}, that is all attributes of the schema, due to the reflexivity of num_sócio and to DF *num_sócio → nome_sócio, bi_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio* (see slide 7.19). Due to *bi_sócio → (nome_sócio, num_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio)*, and reflexivity of bi_sócio, also *{bi_sócio}+ = {bi_sócio, nome_sócio, num_sócio, data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio}*. Thus, the sócio schema is also in the BCNF.

Regarding the filme schema, it is also realized that only the subsets of attributes of this schema that include the attribute *cod_fime* have a closure that includes other attributes, in particular, includes all the attributes of the schema (note the DF *cod_filme→ (nome_filme, ano_filme, preço_dia_filme, dias_sem_multa_filme, multa_dia_filme, cod_género, cod_editora*). All other subsets have a closure limited to the set from which you want to calculate the closure, taking into account the contents of the schema. Therefore, Filme is also in BCNF.

Notice that although *{cod_género}+ ={cod_género, nome_género}* (see the DF cod_género → nome_género), the *nome_género* attribute is not part of the schema and therefore the condition defined in slide 7.38 is verified*: (the closure of α either includes no attributes of Ri-α or includes all attributes of Ri.* Since *α= cod_género* and *Ri = (cod_filme, nome_filme, ano_filme, preço_dia_filme, dias_sem_multa_filme, multa_dia_filme, cod_género, cod_editora).*

As for the *Filme_actor* schema, the only Df applied to it is the one obtained by reflexivity *cod_filme,cod_actor → cod_filme,cod_actor*, which does not violate the condition in 7.38. In fact, Dfs that involve each of the attributes atributos (*cod_filme → nome_filme* and *cod_actor →nome_actor*) do not violate the referred condition. So, *Filme_actor* is in the BCNF. A similar analysis applies to the *Filme_realizador* schema and we will conclude that it is in the BCNF.

Regarding the *Cópia* schema there is no DF that applies to it and that violates the condition in 7.38, as is easily concluded. In fact, *{num_cópia}*+ only contains a *num_cópia* by reflexivity and, regarding the closure of *cod_filme*, all the attributes contained therein that are different from *cod_filme* are not in this schema. So, this schema is therefore in the BCNF.

Regarding the *Aluguer* schema, since it has 4 attributes{*cod_filme, num_cópia, data_aluguer* e *num_sócio*}, there are potentially 2^4 - 1 = 15 subsets whose closures must be considered. However, only the closure {*cod_filme, num_cópia, data_aluguer}*+= *{ cod_filme, num_cópia, data_aluguer}*, obtained based on *(num_copia, cod_filme,_data_aluguer) → num_sócio* (in turn obtained by DF decomposition *(num_copia, cod_filme,__data_aluguer) → (num_sócio, data_devolução, estado_devolução*) requires analysis, since the remaining subsets of attributes have closures limited to the elements of each set. Thus, since the closure *{cod_filme, num_cópia, data_aluguer}*+ contains the complete set of schema attributes, here there is no violation of the condition on slide 7.38 too. The *Aluguer* schema is also in BCNF.

Regarding the Devolução schema, only *(num_copia, cod_filme,_data_aluguer) → (num_sócio, data_devolução, estado_devolução)* requires our analysis, since all other subsets derived from the schema and different from {*num_copia, cod_filme,_data_aluguer*} generate equal closures to all its attributes. Thus, as can easily be seen, the *{num_copia, cod_filme,_data_aluguer}*+ closure contains all the attributes of the schema due to the referred DF and the reflexivity of the attributes to the left of that same DF. Therefore, this schema is also in the BCNF. And so the entire database is in BCNF.

b) Verify that the database preserves the DFs.

DFs will be preserved if *(F_Sócio ∪ F_ Género ∪ F_ Editora ∪ F_Actor ∪ F_ Realizador ∪ F_ Filme ∪ F_Filme_actor ∪ F_ Filme_realizador ∪ F_ Cópia ∪ F_ Aluguer ∪ F_devolução)*+ = *F*+.

where *F_X* is the set of DFs of F+ which applies to schema *X.* Thus, starting with
F_Sócio, the DFs that are applicable to this schema are:
num_sócio → (nome_sócio, bi_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio),
bi_sócio →(nome_sócio, num_sócio,data_nsc_sócio, morada_sócio, tlf_sócio, sexo_sócio)
and all those that are derived by reflexivity from the non-empty 2^9-1=511 subsets formed from these 9 attributes.

Regarding *F_Género*, the DFs that apply to it are the *cod_género → nome_género* and those derived by reflexivity and decomposition*:* (*cod_género, nome_género) → nome_género; (cod_género, nome_género) → cod_género; cod_género→ cod_género e nome_género → nome_género.*

For the remaining sets of DFs, the same work would have to be done. However, it is not difficult to see that the closure of the union of the closures of these sets of DFs is equivalent to the closure F+, since there are no "hidden and unreflected" DFs in the schemas.

Exercise 3:

a) Make sure the database on the topic that your group chose, drawn using the E-R model, is in BCNF. If it is not, make sure it is on 3FN.