Third Normal Form

SAPIENZA
UNIVERSITÀ DI ROMA

The content of the slides is authored by Prof. Maria de Marsico, Sapienza University of Rome
Part of the material is taken from the book: Atzeni, Ceri, Paraboschi, Torlone Basi di dati - Modelli e linguaggi di interrogazione , McGraw-Hill, 2002.

What are the problems with a poorly designed DB?

let's go back to our example of a database containing information about students and exams taken, and start again from the "good" solution found at the end

Hypothesis 3

we have four schemas:

- Student (Matr, TaxC, SurN, Name, DateB, Town)
- Course (C#, Tit, Doc)
- Exam (Matr, C#, DateP, Grade)
- Municipality (Town, Prov)

Hypothesis 3 (considerations)

- Student (Matr, TaxC, SurN, Name, DateB, Town)

- since a matriculation number uniquely identifies a student, each matriculation number corresponds to:
- one tax code (Matr → TaxC)
- one surname (Matr → SurN)
- one name (Matr → Name)
- one date of birth (Matr → DateB)
- one place of birth (Matr → Town)

- so, for a Student instance to be **legal** it must satisfy the functional dependency

Matr → Matr TaxC SurN Name DateB Town

Hypothesis 3 (considerations)

- with similar considerations, we have that an instance of Student to be legal must satisfy the functional dependency

$$\text{TaxC} \rightarrow \text{Matr TaxC SurN Name DateB Town}$$

- therefore both Matr and TaxC are Student **keys**

Hypothesis 3 (considerations)

- on the other hand we can observe that there can be two students with the same surname and different names therefore

- **we can have instances of Student that do not satisfy** the functional dependency

$$SurN \rightarrow Name$$

Student

Matr	TaxC	SurN	Name	DateB	Town
01	…	Rossi	Mario	1/1/1989	Rome
02	…	Bianchi	Paolo	1/1/1989	Rome
03	…	Rossi	Paolo	30/11/1988	Marino
04	…	Neri	Paolo	25/10/1988	Marino

we can have instances of Student that do not satisfy the functional dependency

SurN → Name

SurN → DateB

SurN → Town

Name → SurN

...etc.

Hypothesis 3 (considerations)

we can conclude that **the only non-trivial functional dependencies that must be satisfied by a legal instance of** Student are of the type

$$K \rightarrow X$$

where K contains a key (Matr or TaxC)

we will see that this is a first condition (with **further refinement)** to obtain a definition of Third Normal Form (3NF)

Hypothesis 3 (considerations)

- Exam (Matr, C#, DateP, Grade)

a student can **only pass** the examination of a course once;

therefore, **for each exam (**identified by the student and the course, Matr C#) there is:

- **one DateP**
- **one Grade**

so, **each legal instance of Examination** must satisfy the functional dependency:

$$\text{Matr C\#} \rightarrow \text{DateP Grade}$$

Hypothesis 3 (considerations)

- Exam (Matr, C#, DateP, Grade)

- on the other hand, a student may take different exams on different dates and report different grades
- therefore, there are instances of Exam that do not satisfy one or both of the functional dependencies:

$$\text{Matr} \rightarrow \text{DateP}$$
$$\text{Matr} \rightarrow \text{Vote}$$

Hypothesis 3 (considerations)

- Exam (Matr, C#, DateP, Grade)

- also, the exam for a certain course may be passed by different students on different dates and with different grades
- therefore, there are instances of Exam that do not satisfy one or both of the functional dependencies:

C# → DateP
C# → Rating

therefore, **Matr C#** is a **key** to Exam
(you can verify that it is also **the only** key)

Hypothesis 3 (conclusions)

for each schema:

- Student (**Matr**, TaxC, SurN, Name, DateB, Town)
- Course (**C#**, Tit, Doc)
- Exam (**Matr**, **C#**, DateP, Grade)
- Municipality (**Town**, Prov)

the only non-trivial functional dependencies that must be satisfied by any legal instance are those of type:

$$\mathbf{K} \rightarrow X$$

where K contains a key

Third normal form

(attempt definition) a relation schema is in **3NF** if **the only non-trivial functional dependencies that must be satisfied by any legal instance** are of the type:

$$\mathbf{K} \rightarrow X$$

where:

- **K contains a key**
- **OR**
- **X is contained in a key**

(still a non-formal "version" of 3NF definition)

Third normal form

Definition

- given a relation schema R and a set of functional dependencies F on R, R is in **3NF** if:

$$\forall\, X \rightarrow A \in F^{+},\ A \notin X$$

- A **belongs to a key** (is **prime**) **OR**
- X **contains a key** (it is a **super key**)

Reminder...

we noticed some properties of functional dependencies even before formalizing them with Armstrong's axioms...

Trivial dependencies

given a relation schema R and two non-empty subsets X, Y of R such that $Y \subseteq X$, we have that every instance r of R satisfies the functional dependency $X \rightarrow Y$

R

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a1	b1	c1	d3

X=ABC
Y=AB

$X \rightarrow Y$ is always satisfied

Functional dependencies (properties)

given a relation schema R and a set of functional dependencies F, we have:

$$X \rightarrow Y \in F^{+} \Leftrightarrow \forall A \in Y, X \rightarrow A \in F^{+}$$

$X \rightarrow Y$ must be satisfied by every legal instance of R (it is in F^{+}!)
if $t_1[X]=t_2[X]$ then it must be $t_1[Y]=t_2[Y]$
obviously, if $A \in Y$ and $t_1[A] \neq t_2[A]$, it cannot be $t_1[Y]=t_2[Y]$
conversely, if $\forall\ A \in Y$, $t_1[A]=t_2[A]$, we will have that $t_1[Y]=t_2[Y]$

R

A	B	C	D
a1	b1	c1	d1
a2	b2	c1	d2
a1	b1	c1	d3

$$A \rightarrow BC \in F^{+}$$

$$\Downarrow \Uparrow$$

$$A \rightarrow B \in F^{+}$$

$$A \rightarrow C \in F^{+}$$

Third Normal Form

- Definition

given a relation schema R and a set of functional dependencies F on R, R is in **3NF** if:

- $X \rightarrow A \in F^+$, $A \notin X$
- A **belongs to a key** (is **prime**) **OR**
- X **contains a key** (it is a **super key**)

notes:

- it can be demonstrated that, to check 3NF, we only need to check all the non-trivial dependencies $X \rightarrow A \in F^+$, obtained by decomposing the dependents of the dependencies in F
- and if we replace $X \rightarrow A \in F$, with $X \rightarrow Y \in F$, we don't know what to do if Y contains both prime and non-prime attributes

Example 1

R=ABCD

F={$A \rightarrow B$, $B \rightarrow CD$}

the key is A, in fact for each legal instance (which must satisfy the dependencies in F):

- if $t_1[A]=t_2[A]$ then $t_1[B]=t_2[B]$ and also...
- if $t_1[B]=t_2[B]$ then $t_1[CD]=t_2[CD]$ and so...
- if t_1 $[A]=t_2[A]$ then $t_1[CD]=t_2[CD]$ then $ACD \in F^A$ and by the union rule...
- $ABCD \in F^A$ and we know that $F^A=F^+$ ... so $\mathbf{A \rightarrow R \in F^+}$
- also, A is singleton so it has no subsets, so **A is a key** (we will see later how to find all the key(s) of a schema)
- A is the only key because B does not determine A, and both C and D do not determine other attributes

Example 1

R=ABCD

F={A → B, B → CD}

key: A

- let's check if R is in 3NF:
  - **we only evaluate dependencies in F**
  - A → B is ok (as A is super key)
  - what about B → CD? according to the definition we have to check the dependencies $X \rightarrow A$, with A singleton
  - so, we consider B → C and B → D, that are in F^+ but not in F
  - for both, B is not a super key and both C and D are not prime, so the schema R is not in 3NF

Example 2

R=ABCD

F={AC $\rightarrow$ B, B $\rightarrow$ AD}

the **key is AC**, in fact for each legal instance (which must satisfy the dependencies in F):

- if $t_1[AC]=t_2[AC]$ then $t_1[B]=t_2[B]$ (the dependency must be satisfied) and also...
- if $t_1[B]= t_2[B]$ then $t_1[AD]=t_2[AD]$ (the dependency must be satisfied) and so...
- if $t_1[AC]= t_2[AC]$ then $t_1[AD]= t_2[AD]$ then $AC \rightarrow AD \in F^A$ and, by the union and reflexivity rules
- $AC \rightarrow ABCD \in F^A$ ... and we know that $F^A=F^+$, so $\mathbf{AC \rightarrow R \in F^+}$
- also, A alone does not determine other attributes, so $A \rightarrow R \notin \mathbf{F^+}$, and the same applies to C... so AC is a key
- **BC is another key**
- ABC is a <u>super key</u>

Example 2

R=ABCD

F={AC → B, B → AD}

keys: AC and BC

let's see if R is in 3NF... we only evaluate dependencies in F

- AC → B is ok (AC is super key)
- what about B → AD? according to the definition, we have to check the dependencies X∈A, with A singleton
- ...B → A and B → D are not in F but in F^+
- B → A is ok, because A is part of a key (prime) but B → D violates the 3NF because B is not superkey (it is only part of a key) and D is not part of a key... so the schema R is not in 3NF

Example 2 - Comments

R=ABCD

F={AC → B, B → AD}

if we use an alternative definition

- $X \rightarrow Y \in F^+$, $Y \not\subset X$
- Y belongs to a key (is prime) or
- X contains a key (it is a super key)

how do we evaluate the second condition on BAD since Y=AD, and A is prime (belongs to a key) but D is not?

we should give a more complex definition that involves examining each attribute on the right individually ... and anyway ... we know we can always apply the decomposition

it is simpler and more elegant to use the given definition

Example 3

R=ABCD

F={AB → CD, BC → A, D → AC}

there are three keys, **AB**, **BC**, and **DB**, in fact for each legal instance:

- if $t_1[AB]= t_2[AB]$ then $t_1[CD]=t_2[CD]$
- so, by adding the trivial dependency AB → AB we have that AB → R
- furthermore, A alone does not determine anything
- and the same for B alone
- so, AB is a key

also, by applying the Armstrong's axioms:

- BC → A is in F^A, so, by augmentation we obtain that BC → AB in F^A
- and by transitivity with AB → CD we obtain that BC → CD is in F^A
- finally, by reflexivity and union BC → R is in F^A
- C alone does not determine anything so BC is another key
- D alone does not determine B but, but together with B it becomes a key

Example 3

R=ABCD

F={AB → CD, BC → A, D → AC}

keys: AB, BC, DB

let's see if R is in 3NF

- AB → CD is ok (AB is a superkey)
- BC → A? it is ok (BC is a super key and A is prime)
- ...but what about D → AC?

D is only part of a key, and AC is not part of any key

so, we apply the decomposition (as we are in F^+):

D → A ok, as A is prime!

D → C ok, as C is prime!

the two attributes on the right are part of different keys but they are both prime, so the schema is in 3NF

Third normal form

- Definition

given a relation schema R and a set of functional dependencies F on R, R is in **3NF** if:

$\forall\ X \rightarrow A \in F^{+}, A \notin X$

- A **belongs to a key** (is **prime**) or
- X **contains a key** (it is a **super key**)

- **note**:
- the condition $A \notin X$ is **important;** in fact, for the axiom of **reflexivity**, if $A \in X$ we will always have $X \rightarrow A$ in F^{A} and so in F^{+}, even when A is not prime and X is not superkey; therefore, if we do no exclude this kind of dependencies, then **no schema** would be in 3NF!

Example

- R=AB
- F={A → B}
- the key is obviously A

let's see if R is in 3NF (without the constraint $A \notin X$):

AB contains the key ... OK

$$F^{+} = \{A \rightarrow B,\ AB \rightarrow AB,\ A \rightarrow A,\ B \rightarrow B\ ....\}$$

A is key... OK

$B \rightarrow B$ is a violation because B is not part of the key nor does it contain the key!!!

as these kind of trivial dependencies are always present in F^{+}, they have to be excluded

Third normal form (example: Hypothesis 3)

each schema:

Student (**Matr**, TaxC, SurN, Name, DateB, Town)
Matr → Matr TaxC SurN Name DateB Town
TaxC → Matr TaxC SurN Name DateB Town
Matr → TaxC
TaxC → Matr

Course (**C#**, Tit, Doc)
C# → C# Tit Doc

Exam (**Matr, C#**, DateP, Grade)
Matr C# → Matr C# DateP Rating

Municipality (**Town**, Prov)
Town → Town Prov

is in 3NF

What happens in the other scenarios?

- let's go back to considering the other decomposition hypotheses

Hypothesis 2

the database consists of three schemas:

- Student (Matr, TaxC, SurN, Name, DateB, Town, Province)
- Course (C#, Tit, Doc)
- Exam (Matr, C#, DateP, Grade)

Hypothesis 2 (considerations)

- Student (Matr, TaxC, SurN, Name, DateB, Town, Province)

any Student instance to be **legal** must satisfy functional dependencies:

- Matr → Matr TaxC SurN Name DateB Town Province
- TaxC → Matr TaxC SurN Name DateB Town Province
- Matr → TaxC
- TaxC → Matr

and since each town is located in a single province it must satisfy the functional dependency

- Town → Prov

Hypothesis 2 (considerations)

- Student (Matr, TaxC, SurN, Name, DateB, Town, Province)

let's consider the functional dependency:

$$\text{Town} \rightarrow \text{Province}$$

it's easy to see that:

- Town **is not a key** for Student (there can be multiple students who were born in the same town)
- Province **does not belong to any key** (the only keys are Matr and TaxC)

Hypothesis 2 (conclusions)

the schema:

Student (Matr, TaxC, SurN, Name, DateB, Town, Province)

is not in 3NF

Hypothesis 1

the database consists of a **single** schema:

Curriculum (Matr, TaxC, SurN, Name, DateB, Town, Prov, C#, Tit, Doc, DateP, Vote)

Hypothesis 1 (considerations)

- Curriculum (Matr, TaxC, SurN, Name, DateB, Town, Prov, C#, Tit, Doc, DateP, Vote)

any instance of Curriculum to be **legal** must meet functional dependencies:

- Matr → Matr TaxC SurN Name DateB Town Prov
- TaxC → Matr TaxC SurN Name DateB Town Prov
- TaxC → Matr
- Matr → TaxC
- Town → Prov
- C# → C# Tit Doc
- Matr C# → DataE Vote
- TaxC C# → DataE Vote

Hypothesis 1 (considerations)

so, the (only) keys are:

- **Matr C#**
- **TaxC C#**

.

in fact, any legal instance of Curriculum has to satisfy the functional dependencies:

- Matr C# → Matr TaxC SurN Name DateB Town Prov C# Tit Doc DateP Vote
- TaxC C# → Matr TaxC SurN Name DateB Town Prov C# Tit Doc DateP Vote

Hypothesis 1 (conclusions)

- Curriculum (Matr, TaxC, SurN, Name, DateB, Town, Prov, C#, Tit, Doc, DateP, Vote)

consider the functional dependency

- Matr → SurN

given that:

- Matr is not a key for Curriculum (a student may have taken more than one exam - remember that for this schema the keys are Matr C# and TaxC C#)
- SurN does not belong to any key
-

Curriculum is not in 3NF

Note: it is enough to identify just the first dependency that violates the conditions for 3NF

Partial dependencies

- Curriculum (**Matr**, TaxC, SurN, Name, DateB, Town, Prov, **C#**, Tit, Doc, DateP, Vote)

any matriculation number identifies a particular surname:

- Matr → SurN

so, any pair consisting of a matriculation number and a course code identifies a surname: Matr C# → SurN

- the depenency Matr C# → SurN is a consequence of the dependency Matr → SurN
- Matr → SurN is called **partial dependency**

Transitive dependencies

- Student (**Matr**, TaxC, SurN, Name, DateB, Town, Province)

- a matriculation number identifies only one town of birth: Matr → Town
- a town is located in only one province: Town → Prov

conclusion:

- a matriculation number corresponds to only one province: Matr → Prov

- the functional dependency Matr → Prov is a consequence of the two functional dependencies Matr → Town and Town → Prov
- Town → Prov is called **transitive dependency**

Alternative definition of 3NF

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- Definition
- given a schema R and a set of functional dependencies F on R, R is in 3NF if and only if there are neither partial dependencies nor transitive dependencies in F