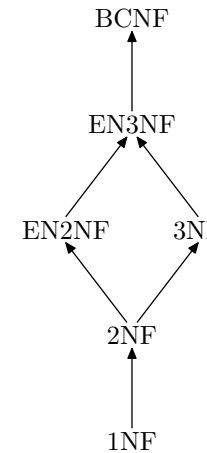


Lecture 23: A BCNF Example, and more!

Why 3NF is sometimes not enough

Today is mostly about an example motivating BCNF, but there will be a couple of detours too.

- functional dependency inference
- “lossless” decomposition
- modelling the example with UML, and translating



- the arrows show increasing strength
- today we will see an example that is in EN3NF, but not BCNF
- exercise: find a relation  $R$  and functional dependencies  $F$  that
  - is in 3NF but not EN2NF
  - is in EN2NF but not 3NF

An Example Motivating BCNF

This example is from [E&N §10.5, Figure 10.13].

- There are *students*, *courses* and *instructors*.
- We are not interested in their properties, only their relationships.
- Therefore, we can represent them as attributes.
- Initial design just has an attribute for each.

student	course	instructor
u123456	Database	O'Keefe
⋮	⋮	⋮

Functional Dependencies of the Example

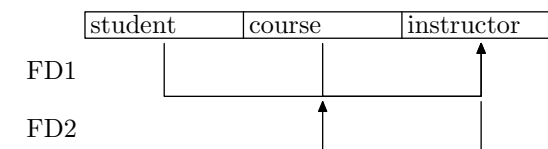
$$FD1 : \{student, course\} \longrightarrow instructor$$

A student can not be taking the same course with two different instructors.

$$FD2 : instructor \longrightarrow course$$

Instructors can take no more than one course.

Shown another way, the functional dependencies are



## Functional Dependency Inference

Note that  $\{student, course\} \longrightarrow instructor$  makes  $\{student, course\}$  a candidate key. (do you agree?)

Is  $\{instructor, student\}$  also a candidate key?

Yes, because *instructor* determines *course* (FD2).

More formally, [E&N §10.2] introduces some rules by which we can deduce functional dependencies from other functional dependencies.

(This is like a logic of functional dependencies, see lecture 16)

## FD Inference - Reflexivity

The “reflexive rule” IR1, is

if  $X \supseteq Y$  then  $X \longrightarrow Y$

Therefore we have

$\{instructor, student\} \longrightarrow \{instructor, student\}$

## FD Inference - Augmentation

One of the rules given is the “augmentation rule” IR2 (footnote version)

$X \longrightarrow Y \models XZ \longrightarrow Y$

Therefore we have

$instructor \longrightarrow course \models \{instructor, student\} \longrightarrow course$

because we have FD2 on the left, we also have

$\{instructor, student\} \longrightarrow course$

## FD Inference - Additivity

The “Additive rule” IR5, is

$\{X \longrightarrow Y, X \longrightarrow Z\} \models X \longrightarrow YZ$

which takes us from

$\{instructor, student\} \longrightarrow \{instructor, student\}$

$\{instructor, student\} \longrightarrow course$

to

$\{instructor, student\} \longrightarrow \{instructor, student, course\}$

So, yes,  $\{instructor, student\}$  is a candidate key.

## Sound and Complete

A formal logic has semantics and a deductive system.

- semantics defines what statements are true in what situations
- deductive system defines proofs from a set of premise statements to a conclusion statement
- the deductive system is *sound* if every provable argument is valid
- the deductive system is *complete* if every valid argument is provable

We have statements (functional dependencies), situations (admissible relations) and a deductive system (rules of inference for FDs).

The first 3 rules give a sound and complete deductive system [E&N §10.2.2]

## Back to the Example

All attributes are prime, because  $\{student, course\}$  and  $\{instructor, student\}$  are both candidate keys.

Recall the stronger forms of 2NF, 3NF:

**EN2NF** 1NF & every non-prime attribute is fully functionally dependent on every key

**EN3NF** EN2NF & every non-prime attribute is non-transitively dependent on every key

If every attribute of a relation is prime, then it is in EN3NF. So, our example is in EN3NF.

## Equivalent Sets of Functional Dependencies

We write  $F^+$  for all the functional dependencies that are implied by those in  $F$ . This is called the *closure* of  $F$ .

(Mathematical people - compare with topology!)

It is possible for  $F^+ = G^+$  even if  $F \neq G$ .

When  $F^+ = G^+$ , we say that  $F$  and  $G$  are *equivalent*.

Sometimes we “lose” functional dependencies when we decompose relations. Sometimes we can avoid this by switching to an equivalent set of functional dependencies!

## Its EN3NF, but is it any good?

We can choose  $\{student, instructor\}$  or  $\{student, course\}$  as the primary key for this relation.

Either way, it is possible to violate

$FD2 : instructor \longrightarrow course$

student	course	instructor
u123456	Comp2400	O'Keefe
u234567	Comp6240	O'Keefe
⋮	⋮	⋮

## Decomposition Choices

- there are three ways we can break this down into 2 relations
  - 1  $\{student, instructor\}, \{student, course\}$
  - 2  $\{course, instructor\}, \{course, student\}$
  - 3  $\{instructor, course\}, \{instructor, student\}$
- they all “lose” FD1, by putting its attributes into separate relations
- the first also loses FD2 by separating *course* and *instructor*
- the first two also lose information
- we want to be able to recover the original, undecomposed table by a natural join \*

## The Stupidest Choice

$R$

student	course	instructor
u123456	Database	O'Keefe
u123456	Philosophy	Descartes

$\pi_X(R)$

student	instructor
u123456	O'Keefe
u123456	Descartes

$\pi_Y(R)$

student	course
u123456	Database
u123456	Philosophy

$\pi_X(R) * \pi_Y(R)$

student	course	instructor
u123456	Database	O'Keefe
u123456	Database	Descartes
u123456	Philosophy	O'Keefe
u123456	Philosophy	Descartes

## Still Pretty Dumb

$R$

student	course	instructor
u123456	Philosophy	Hegel
u234567	Philosophy	Descartes

$\pi_X(R)$

course	instructor
Philosophy	Hegel
Philosophy	Descartes

$\pi_Y(R)$

student	course
u123456	Philosophy
u234567	Philosophy

$\pi_X(R) * \pi_Y(R)$

student	course	instructor
u123456	Philosophy	Hegel
u234567	Philosophy	Descartes
u123456	Philosophy	Hegel
u234567	Philosophy	Descartes

## Best of a Bad Bunch?

$R$

student	course	instructor
u123456	Philosophy	Hegel
u234567	Philosophy	Descartes
u234567	Database	O'Keefe

$\pi_X(R)$

course	instructor
Philosophy	Hegel
Philosophy	Descartes
Database	O'Keefe

$\pi_Y(R)$

student	instructor
u123456	Hegel
u234567	Descartes
u234567	O'Keefe

$\pi_X(R) * \pi_Y(R)$

student	course	instructor
u123456	Philosophy	Hegel
u234567	Philosophy	Descartes
u234567	Database	O'Keefe

## Why was that one better?

- each decomposition has a common attribute in both projections
- the join is on that attribute
- if both projections allow multiple rows with the same value for that attribute, we are in trouble
- but if the attribute is a key in one projection, there will be no loss of information (ie gain of non-information)
- in the last example decomposition, the common attribute, *instructor* is a key in its projection because of FD2:  $instructor \longrightarrow course$
- this is why foreign key constraints must reference keys [E&N §5.2.4]

## Lost and Found

We have “lost” FD1:  $\{student, course\} \longrightarrow instructor$ , because none of the projections has all 3 attributes.

The point of FD1 was that a student can not be in the same course twice with different instructors.

But we have captured it in a way, because we can not represent that situation with the schema we got.

That is, its impossible to get following relation by joining instances of (*instructor*, *course*) and (*instructor*, *student*).

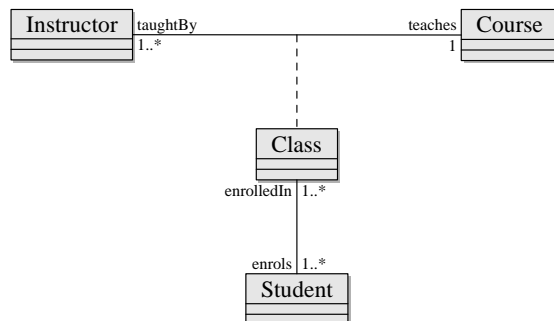
student	course	instructor
u123456	Philosophy	Hegel
u123456	Philosophy	Descartes

## Class Diagram for BCNF Example

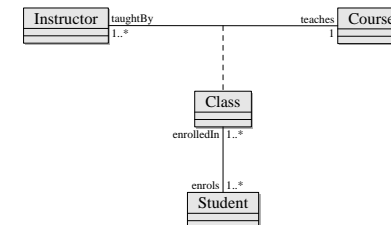
Modelling this example with UML will provide

- some insight into the example
- more familiarity with UML modelling and translation
- food for thought on modelling, normalisation and representation

Here’s my suggestion



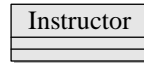
## Model Observations



- there is no mention of Class in the text, but imho, there should be, because that’s what a student really enrolls in
- maybe it should show instructor teaches 0..1 courses?
- note that we have *no attributes*
- let’s work with this no-attribute UML subset!
- it will allow us to develop a pretty funky translation

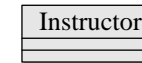
## How do we Translate This?

Note, we are now developing a *different* translation, from a *different* UML subset.



- artificial “identifier” attribute is implicit
- we would usually declare this as say *instructorID* : *id* for some domain *id* of identifiers
- but since we have no UML attributes, we can just call it *instructor* : *id*
- `varchar(20)` is just as good an identifier as `int` or `whatever`

## Classes Become Relations



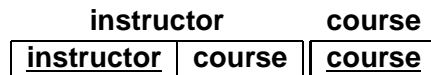
```
CREATE TABLE instructor (
instructor id;
);
```

Instructor
<b>instructor</b>
Descartes
O'Keefe
⋮

## How do we Translate This?



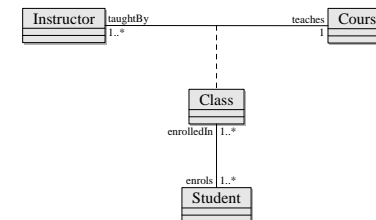
The standard association translation, yields



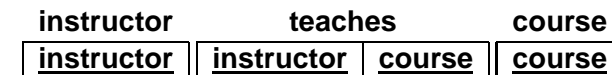
with

- foreign key *instructor(course)* referencing *course(course)*

## How do we Translate This?



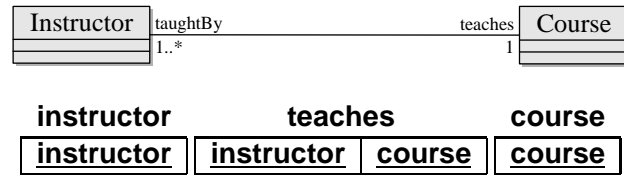
But, because the association has an association class, which participates in another association with \* at the far end, it requires its own relation.



with

- foreign key *course* referencing *course(course)*
- foreign key *instructor* referencing *instructor(instructor)*

## How do we Translate This?



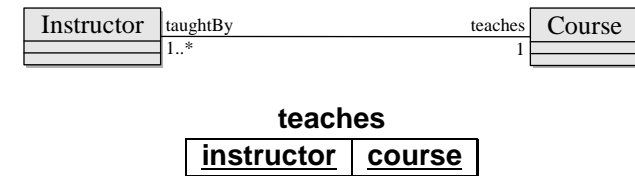
- *but*, notice that both association end multiplicities have lower bound 1
- therefore

$$course = \pi_{course}(teaches)$$

$$instructor = \pi_{instructor}(teaches)$$

- that is, we can throw the *course* and *instructor* tables away

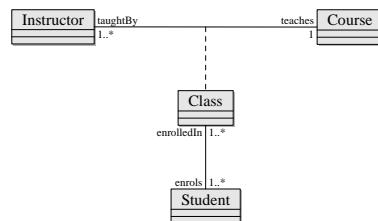
## How do we Translate This?



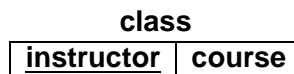
- *but*, notice that this looks pretty damned similar to the table we had originally for *instructor*
- and that one is faithful to the right hand side multiplicity of 1



## How do we Translate This?

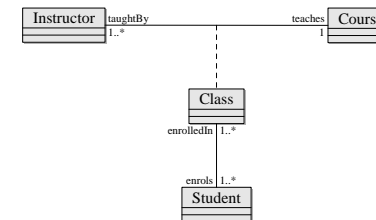


Since that captures the association, which “is” (!?!) the association class ...

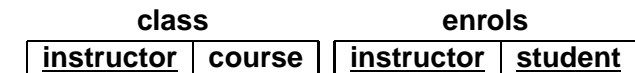


We translate Class, and everything above it by this relation.

## How do we Translate This?



The *enrols* association is many to many, so it must become a relation ...



- foreign key  $enrols(instructor)$  references  $class(instructor)$
- as before, there is no need to store the *student* relation, because  $\pi_{student}(enrols) = student$
- (this is true even though each student may be enrolled in many classes, and thus  $|student| \leq |enrols|$ )

## What did we just do?

- described the problem “world” using a model
- exploited special features of the language fragment used to develop a very economical translation
  - we did not explicitly formulate the translation procedure
  - but there is one there which applies to any model with only attributeless classes, associations and association classes
  - it is nicest when multiplicities have lower bound 1
  - exercise: write down the procedure
- the result is the “desirable” non-additive BCNF schema we saw before
- Is normalisation only needed to repair bad or absent modelling?