

25 YEARS ANNIVERSARY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# IT3090E - Databases

Chapter 5: Database design – bottom-up approach  
*Functional dependencies*

Muriel VISANI

[murielv@soict.hust.edu.vn](mailto:murielv@soict.hust.edu.vn)

# Designing a database

- To design a database, one can start from a data **state**
  - List of data that the database users need

## ⇒Bottom-up approach

- The states can be obtained by
  - The analysis of the existing system (flat files or other ...)
  - An interview with DB users to determine which states they would need

# Designing a database

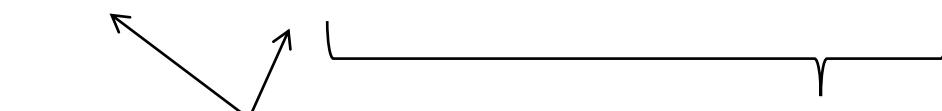
- Example :
  - In the case of a supermarket, you can start with the information on the receipts
  - In the case of a library, you can start from the excel files necessary for the library to function properly
  - In the case of the school service, you can start from the planning sheets...
  -

# Designing a database - example 1

- How do I organize a movies collection?
- How to describe a movie??
  - Title, category (sci-fi, comedy...), type (DVD, BluRay, Divx)
  - Lead actor, director

–

<b>Id</b>	<b>Title</b>	<b>Cat</b>	<b>Type</b>	<b>Director</b>	<b>Actor</b>
1	Indiana Jones	advent	dvd	S. Spielberg	H. Ford
2	Indiana Jones II	advent	dvd	S. Spielberg	H. Ford
3	Indiana Jones III	advent	dvd	S. Spielberg	H. Ford
4	Indi Jones III	advent	dvd	S. Spielberg	S. Connery



**Title** cannot be PK -> we need  
to add an identifier

The values of the attributes  
**Title, Cat, Type, Director**  
are the same for multiple records

What if we'd like to...

- Write « Harrison Ford » ?
- Modify the title « Indi III » ?
- Delete Indi III ?

**We need to do the same thing  
multiple times -> risk of error**

# Designing a database - example 2

- How to organize the data?
- Example: Paleontology
  - Information about dinosaurs
    - Name, Era, Beginning, End, Ecosystem, Diet, Weight
  - Information on eras (cretaceous, jurassic...)
    - era (cretaceous, jurassic...), Beginning, End

*D (Name, Era, Beginning, End, Ecosystem, Diet, Weight)*

Allosaurus	Jurassic	200	145	Ground	carnivore	2.09
Ptéranodon	cretaceous	145	65	Air	carnivore	NULL
Brontosaurus	Jurassic	200	149	Lake	herbivorous	32.48
Tyrannosaurus	Cretaceous	145	65	Ground	carnivore	6.89

# Designing a database - example 2

- Problems :
  - Redundencies: information about Eras (Beginning, end)
  - Inconsistencies: Two timeframes for Jurassic; Cretaceous or Cretaceous?
  - Incomplete: No information on the Triassic period if no dinosaur in the DB
  - Difficulties to update or insert new data
  - D (Name, Era, Beginning, End, Ecosystem, Diet, Weight)

Allosaurus	Jurassic	200	145	Ground	carnivore	2.09
Ptéranodon	cretaceous	145	65	Air	carnivore	NULL
Brontosaurus	Jurassic	200	149	Lake	herbivorous	32.48
Tyrannosaurus	Cretaceous	145	65	Ground	carnivore	6.89

# Decomposition

- How can we avoid these problems?
  - By breaking down D into 2 tables + 1 foreign key

Dinosaur (Name, Era#, Ecosystem, Diet, Weight)

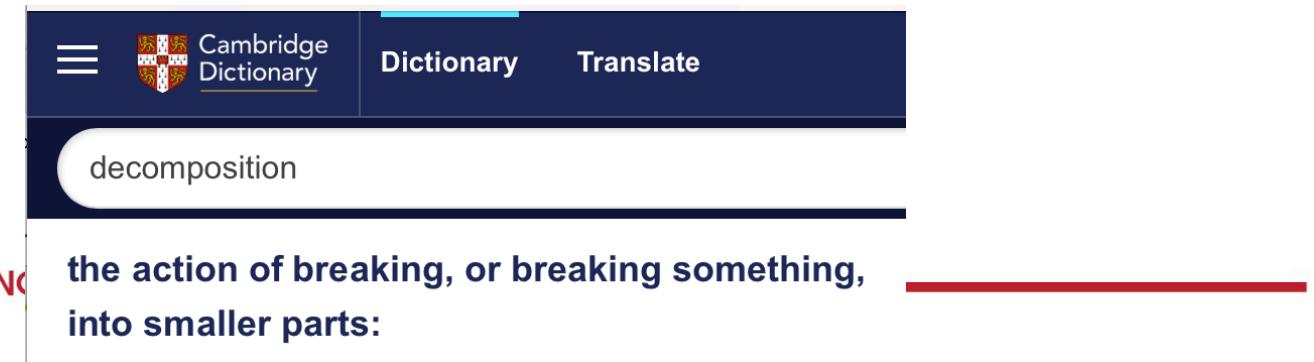
Allosaurus	Jurassic	Ground	carnivore	2.09
Ptéranodon	cretaceous	Air	carnivore	NULL
Brontosaurus	Jurassic	Lake	herbivorous	32.48
Tyrannosaurus	cretaceous	Ground	carnivore	6.89

Era (Era, Beginning, End)

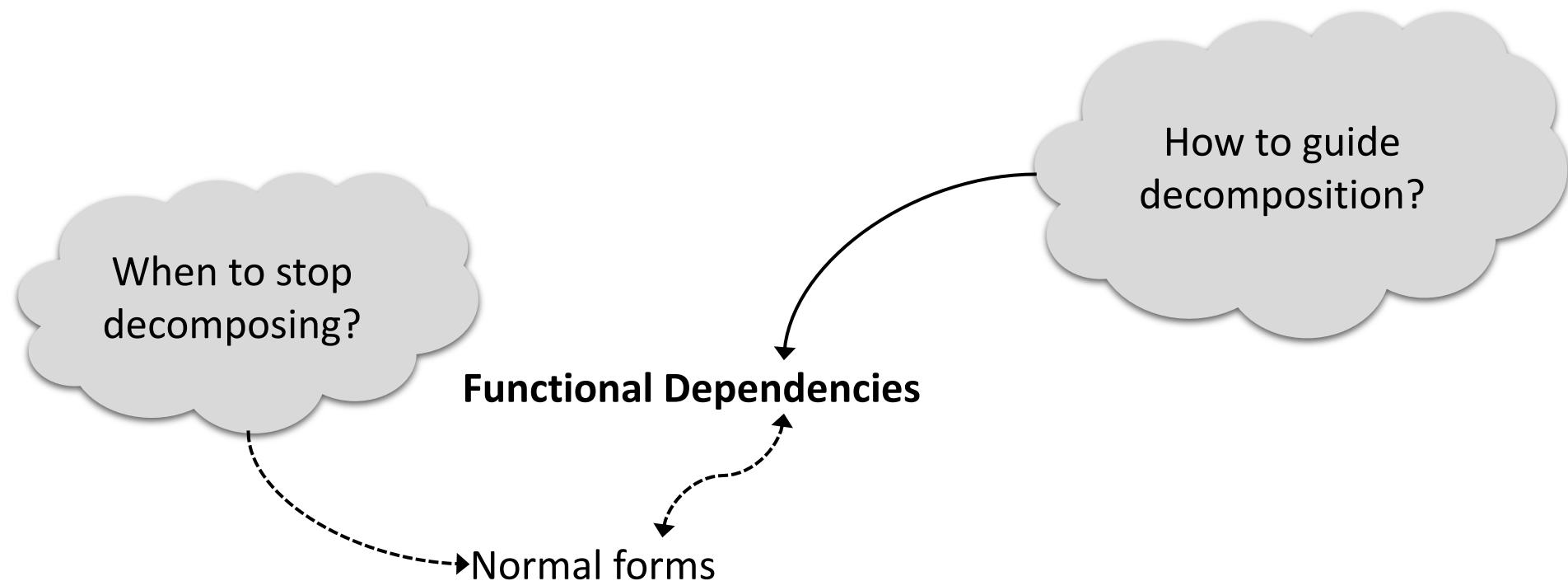
Jurassic	200	145
cretaceous	145	65
triassic	230	200

# Decomposition

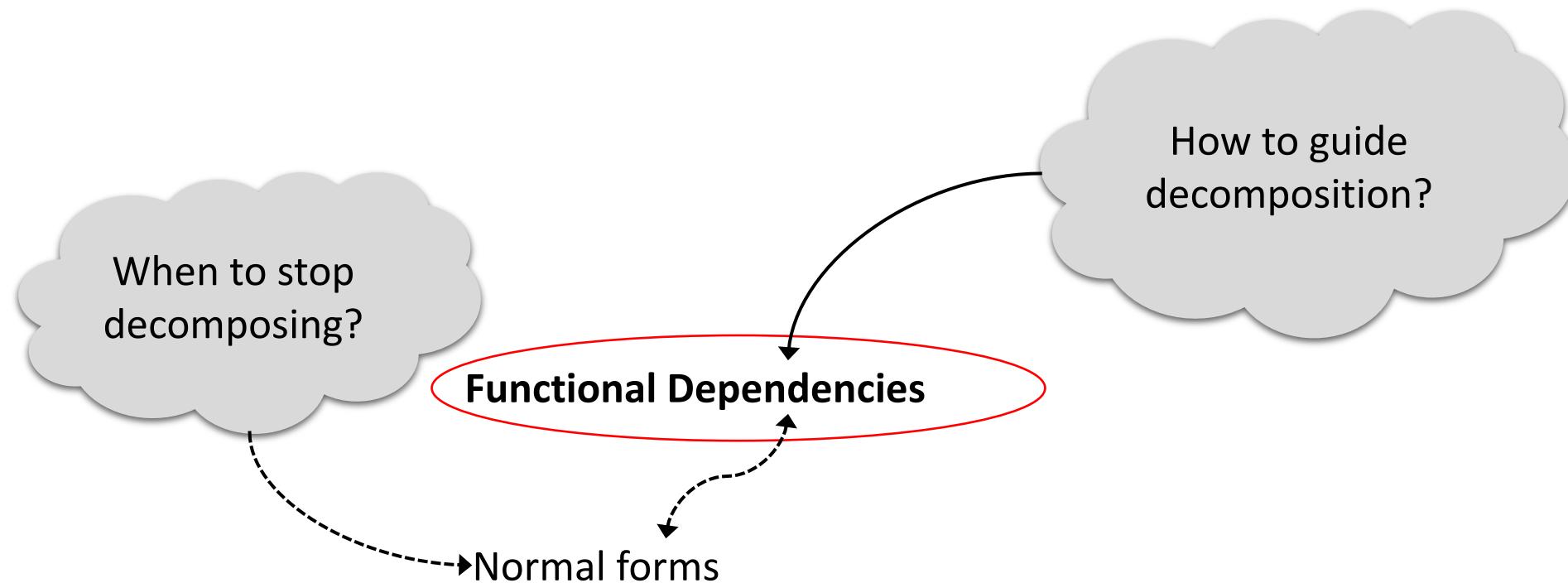
- Designing a « good » relational database
  - No redundancy
  - Easier update
  - Better-assured consistency
  - Easier understanding
  -
- Principle → break down (**decompose**) one table into several tables
  - How to do that?
  - When to stop?



# Decomposition



# Functional Dependencies (FDs)



# Functional Dependencies (FDs)

- Functional Dependencies (FDs) are an expression of:
  - The semantic links between attributes
  - Our choices for representing reality
  -

$$A \rightarrow B$$

**A determines B ( $\Leftrightarrow$  B is functionally dependent of A)**  
If and only if

**The value of B ONLY depends on the value of A ( $\Leftrightarrow$  for each value of A, the value of B is always the same)**

A is called the **determinant**, B is called the **dependent**

*In general, A is a primary key (but not always)*

# Functional Dependencies (FDs) - examples

- Tax Id → Name?
- Licence plate → Model?
- Colour → Licence plate?

# Functional Dependencies (FDs)

- Watch out for the direction of the DF!!
  - The right side (dependent) depends on the left side (determinant)
- Example: if a student has only one grade per course
  - Nguyen Tú, StudentID 100, got A+ in Law and Computer Science
  - **StudentID, Grade → Course?**
    - 100, A+ → Law ; 100, A+ → Computer Science
  - **StudentID, Course → Grade?**
    - 100, Law → A+ ; 100, Computer Science → A+

# Functional Dependencies (FDs)

- **Example:** R (OrderID, OrderDetailID, ProductID, Order\_Date, Quantity)
  - OrderNo: identifier for the order
  - OrderDetailID: identifier for the order line
  - ProductID: identifier for the product ordered
  - Quantity: ordered quantity (for that product)
  - 1 order line / product ordered, multiple order lines / order*
  - Order\_date: date of the order

Example of corresponding state: receipt

Order n° 10123			
16/04/2021			
Product	Unit Price	Qty	Total Price
USB Key	9.90	2	19.8
USB hub	16.98	1	16.98

**N.B.** The state does not necessarily contain all the attributes we need in the database (especially PKs...)

# Functional Dependencies (FDs)

- Example: **R (OrderID, OrderLineID, ProductID, Order\_Date, Quantity)**

OrderID: identifier for the order

ProductID: identifier for the product ordered

OrderLineID: identifier for the order line

Quantity: ordered quantity (for that product)

*1 order line / product ordered, multiple order lines / order*

Order\_Date: date of the order

Example of corresponding state: receipt

Order n° 10123			
16/04/2021			
Product	Unit Price	Qty	Total Price
USB Key	9.90	2	19.8
USB hub	16.98	1	16.98

**Questions - is it true that:**

$\text{OrderID} \rightarrow \text{Order\_Date}$  ??

$\text{OrderID} \rightarrow \text{ProductID}$  ??

$\text{OrderID} \rightarrow \text{OrderLineID}$  ??

$\text{OrderID}, \text{OrderLineID} \rightarrow \text{ProductID}, \text{Quantity}??$

# Identifying functional dependencies

- Looking at the data in the DB is **not enough!!!**
  - A FD is not linked to the specific state of the database at a given instant
    - A FD is more like a general rule
  - **Beware:**  $A \rightarrow B$  can be verified in practice, without it being a DF (general rule)
    - Example: from this extract, one might think that Title  $\rightarrow$  Type...
    - Until we get Indiana Jones in blue-ray...

<b>Id</b>	<b>Title</b>	<b>Cat</b>	<b>Type</b>	<b>Director</b>	<b>Actor</b>
1	Indiana Jones	advent	dvd	S. Spielberg	H. Ford
2	Indiana Jones II	advent	dvd	S. Spielberg	H. Ford
3	Indiana Jones III	advent	dvd	S. Spielberg	H. Ford
4	Indi Jones III	advent	dvd	S. Spielberg	S. Connery

# Identifying functional dependencies

- Therefore, it is difficult to automatize the identification of FDs...
  - A FD is a general rule that humans can only identify if they understand the DB contents...
  - Usually, you'll need an interview with the DB users to really identify functional dependencies...

# Definitions and properties

- The three Armstrong's **axioms** are the following, where:
  - $R = \{A_1, A_2, \dots, A_n\}$ ,  $X, Y, Z, W$  are subsets of  $R$
  - $XY$  denotes  $X \cup Y$
- **Reflexivity**
  - If  $Y \subseteq X$  then  $X \rightarrow Y$
- **Augmentation**
  - If  $X \rightarrow Y$  then  $XZ \rightarrow YZ$
- **Transitivity**
  - If  $X \rightarrow Y$ ,  $Y \rightarrow Z$  then  $X \rightarrow Z$

# Definitions and properties

- Armstrong's **secondary rules** are the following:
  - If  $X \rightarrow Y$ ,  $X \rightarrow Z$  then  $X \rightarrow YZ$ .
- Pseudo-transitivity
  - If  $X \rightarrow Y$ ,  $WY \rightarrow Z$  then  $XW \rightarrow Z$ .
- Decomposition
  - If  $X \rightarrow Y$ ,  $Z \subseteq Y$  then  $X \rightarrow Z$

# Exercises

- Consider the following set of FDs on the attributes A,B,C,D,E,F:

$$A \rightarrow B$$

$$A \rightarrow C$$

$$CD \rightarrow E$$

$$CD \rightarrow F$$

$$B \rightarrow E$$

- Exercise 1: Prove  $CD \rightarrow EF$  by applying Armstrong's Axioms (reflexivity, augmentation, and transitivity)

# Exercises

- Solution to exercise 1:

# Exercises

- Consider the following set of FDs on the attributes A,B,C,D,E,F:

$$A \rightarrow B$$

$$A \rightarrow C$$

$$CD \rightarrow E$$

$$CD \rightarrow F$$

$$B \rightarrow E$$

- Exercise 2: Prove  $AD \rightarrow EF$  by repeatedly applying Armstrong's Axioms

# Exercises

- Solution to exercise 2:

# Definitions and properties

- Trivial FDs:
  - A FD  $X \rightarrow Y$  is **trivial** if  $X \supseteq Y$ 
    - In particular,  $X \rightarrow X$  and  $\{X, Y\} \rightarrow X$  are always true
- Partial FDs:
  - A FD  $\{X, Y\} \rightarrow Z$  is **partial** if  $X \rightarrow Z$ 
    - (In other words,  $Y$  is « useless » in that FD)
- Elementary FDs:
  - A FD  $X \rightarrow Y$  is **elementary** if it is nor trivial nor partial
  - If  $X \rightarrow Y$  is an elementary FD, then  $X$  is a **candidate key** for the table  $\{X, Y\}$

# Definitions and properties

- In Chapter 5bis (on Teams), you'll find 2 algorithms to find the minimal key of a table
  - Based on the notion of closure of a set of attributes
- Please read it carefully and ask me questions if needed
- In this lecture, we will focus on more hand—on experience for database design: decomposing the tables and until we reach some normal form
  - For finding the keys, we will rely mainly on common sense and semantics

# Decomposition - example

Item	Ref	Qty	Supplier	ZIP_code	City
Umbrella	501	50	Labalaine	75000	Paris
Hat	530	100	Lemelon	17000	La Rochelle
Rain jacket	510	100	Labaleine	75000	Paris

- The attributes ZIP\_code and City are dependent on the attribute supplier:
  - Supplier → ZIP\_code, City

# Decomposition - example

- Therefore, we can decompose the initial relation:
  - Supplier → ZIP\_code, City

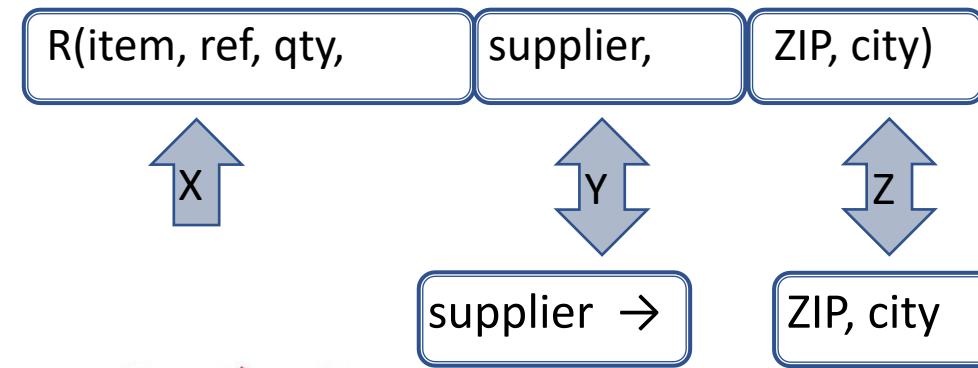
Item	Ref	Qty	Supplier	ZIP_code	City
Umbrella	501	50	Labalaine	75000	Paris
Hat	530	100	Lemelon	17000	La Rochelle
Rain jacket	510	100	Labaleine	75000	Paris

Item	Ref	Qty	Supplier#
Umbrella	501	50	Labalaine
Hat	530	100	Lemelon
Rain jacket	510	100	Labaleine

Supplier	ZIP_code	City
Labalaine	75000	Paris
Lemelon	17000	La Rochelle

# Decomposition – general rule

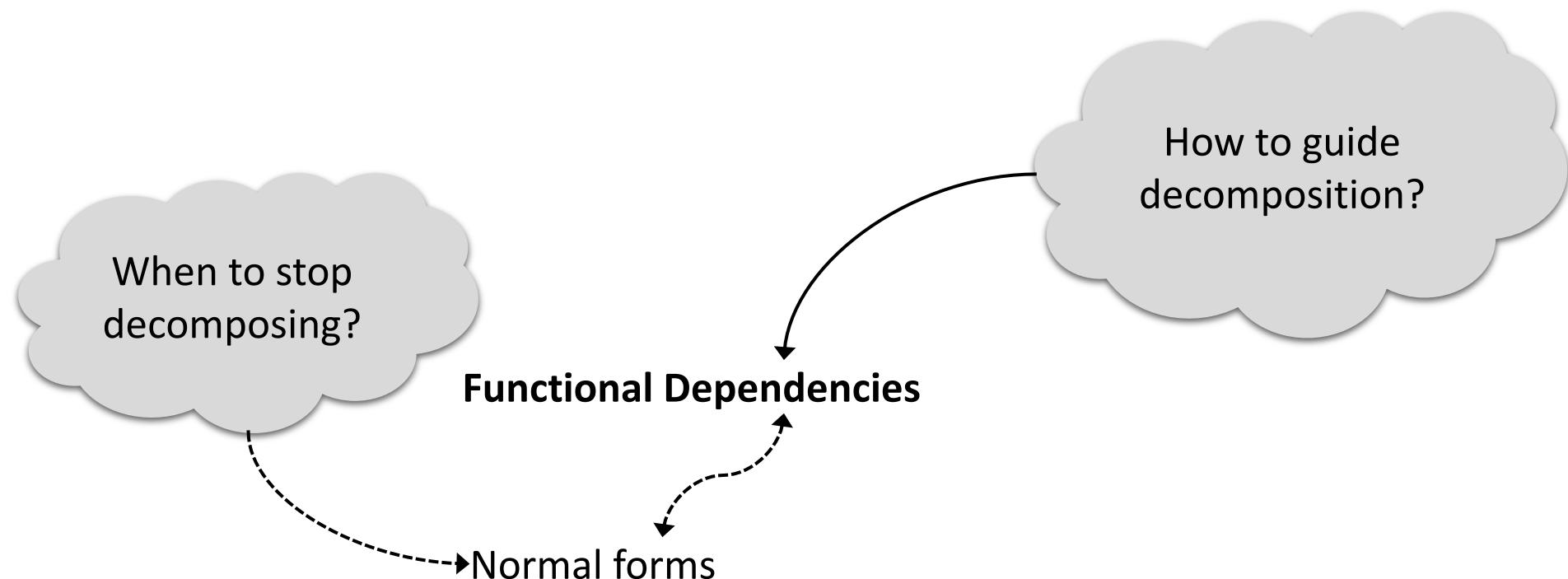
- General rule:
  - If  $R(X, Y, Z)$  is a relation
  - And if  $Y \rightarrow Z$  is a FD of  $R$
  - Then  $R(X, Y, Z) = R(X, Y) * R(Y, Z)$
- In other words, the right hand-side of the FD can be taken out of the initial relation  $R$



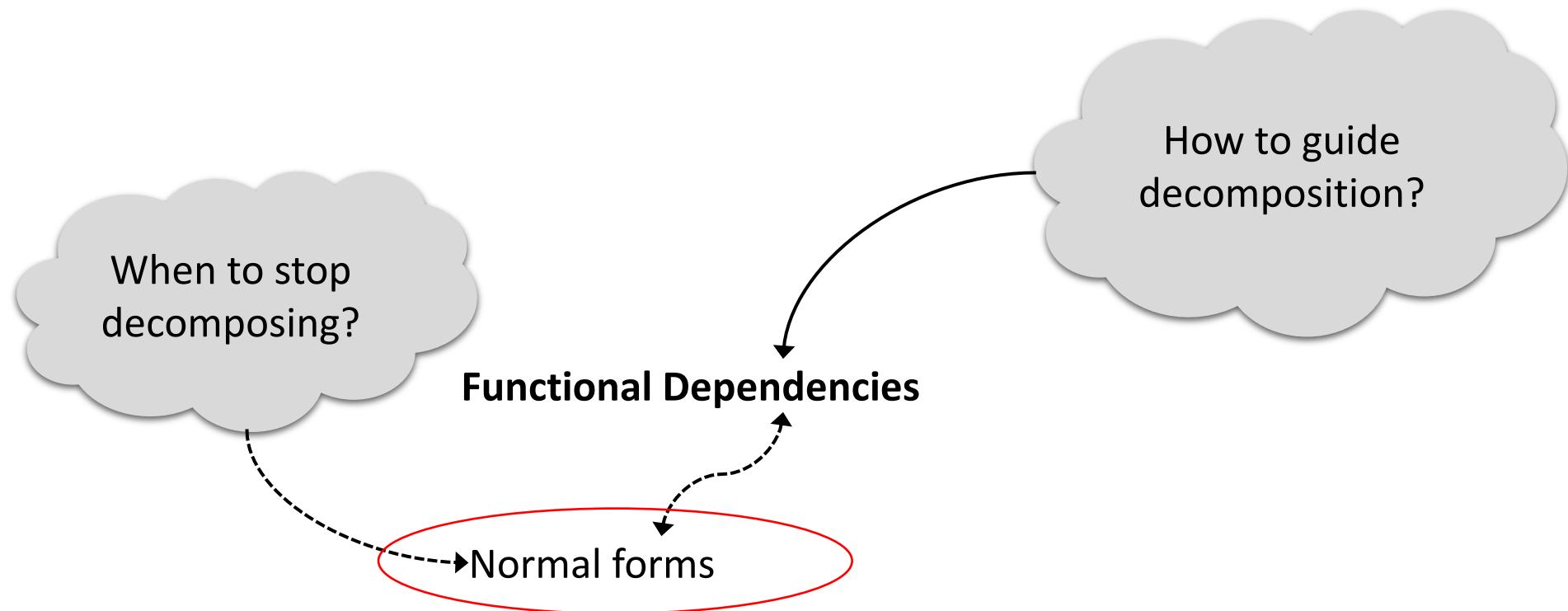
# Decomposition – some comments

- If we decompose a table following a DF, then we don't lose any information
  - The **join** between the two tables contain the same information as the initial table
- One can always decompose a table when there exists **at least one** DF
- One cannot decompose a table when there exists **no** DF

# Decomposition



# Normalization

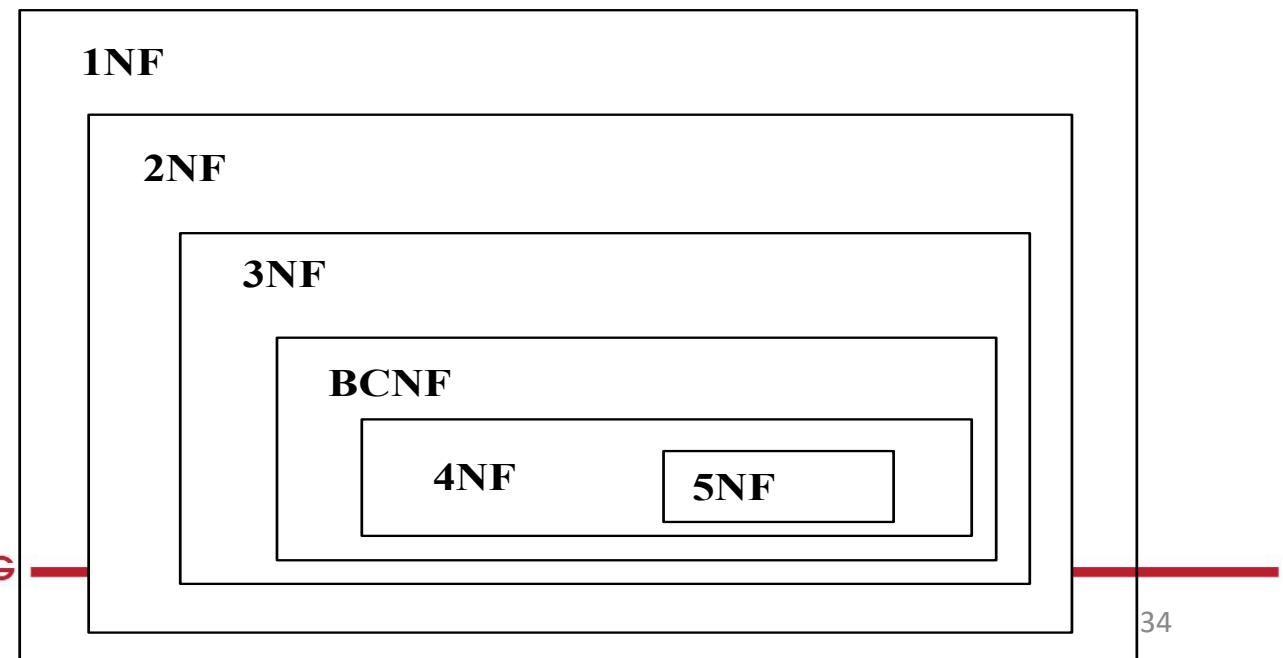


# Why normalizing?

- Normalizing a relational schema is replacing it with an equivalent schema where all tables verify certain properties
  - These properties are based on the analysis of FDs inside each table
- Normalizing is useful to:
  - Limit data redundancies
  - Minimize storage space
  - Limiting data loss
  - Limit inconsistencies
  - Improve processing performances
  - Avoid any difficulty (potential source of errors) during updates

# Normal forms

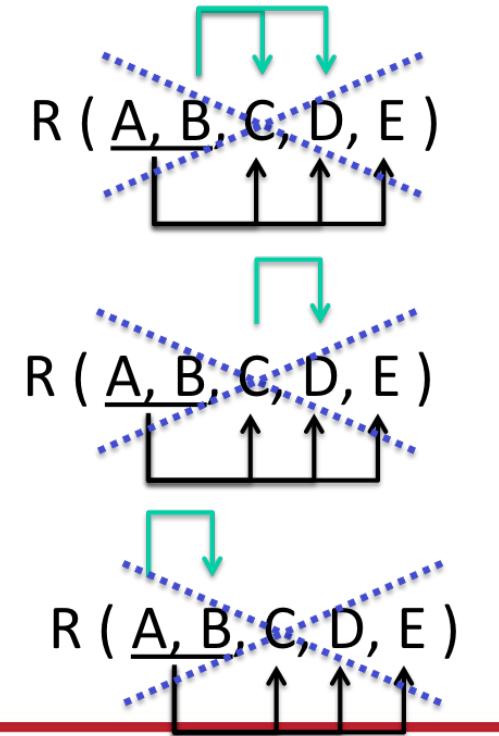
- There exist different levels of normalization
  - 1st, 2<sup>nd</sup> and 3rd Normal forms (1NF, 2NF, 3NF)
  - Boyce-Codd Normal Form (BCNF)
  - The 4th, 5th normal forms are complex (we'll not study them today)



# Normal forms

- 1NF :
  - No calculated or non-atomic attribute
- 2NF :
  - 1NF + No Partial Dependency (PD)
    - PD: when a **non-prime attribute** (attributes which are not part of any candidate key) is dependent on a subset of a candidate key
- 3NF :
  - 2NF + no Transitive Dependency for non-prime attributes
    - TD happens when a non-prime attribute depends on another non-prime attribute
- BCNF :
  - 3NF + in every FD, the determinant is a **super-key**
    - Super-key: set of attributes which identifies uniquely the attributes in the relation (not necessarily minimal)

*Illustrations (specific cases)*



# How to normalize a schema?

- Basically, it is made by:
  - Initialization: putting all attributes in one big table R
  - Identification of all the elementary FDs in the table R
  - Decomposing recursively the table R by applying the FDs...
  - ... until we reach the level of normalization that we wish to get
    - Usually, 3NF is enough...

# Decomposition algorithm

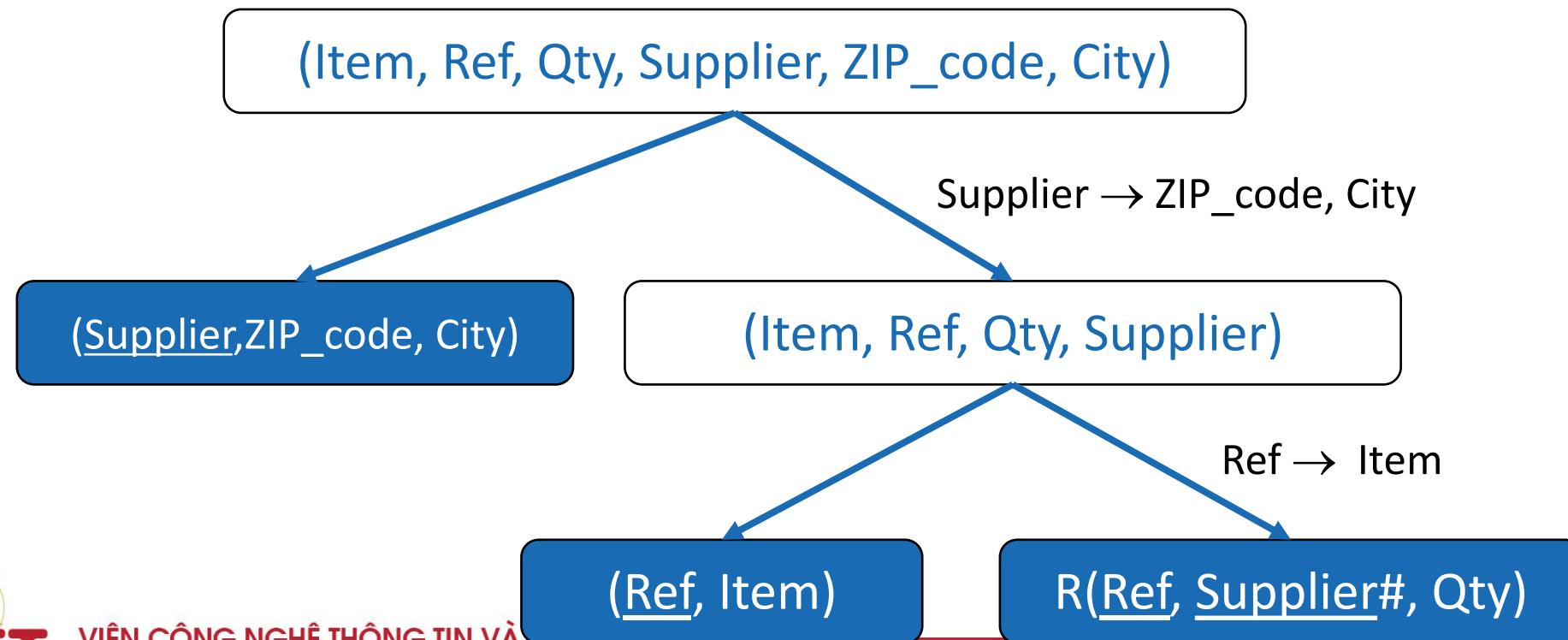
- Input:
  - A relation R to decompose
  - A set of FDs
- Output:
  - Multiple tables which join gives R
- Principle:
  - Recursive algorithm
  - Equivalent to calculating a binary tree which root is R
  - The tree leaves give the final result

# Decomposition algorithm

- Recursive decomposition step
- Current node: a relation  $R(T)$ 
  - Choose a DF  $Y \rightarrow Z$  which attributes are in  $R$ 
    - $\Rightarrow T$  can be decomposed into  $T = X \cup Y \cup Z$
  - Add the relation  $R(Y, Z)$  as the left child of  $R$
  - Add the relation  $R(X, Y)$  as the right child of  $R$
  - Recursively decompose  $R(X, Y)$  and  $R(Y, Z)$

# Example

- Input:
  - Table: (Item, Ref, Qty, Supplier, ZIP\_code, City)
  - FDs: Supplier → ZIP\_code, City; Ref → Item



# Normal forms

- Exercise 3:
  - Given the relation R (A, B, C, D, E) and the following FD:
    1.  $A \rightarrow B$
    2.  $BC \rightarrow E$
    3.  $DE \rightarrow A$

Question 1: give the candidate key(s) of R (see slide 25)

Solution:

-

# Normal forms

- Exercise 3:
  - Given the relation R (A, B, C, D, E) and the following FD:
    1.  $A \rightarrow B$
    2.  $BC \rightarrow E$
    3.  $DE \rightarrow A$

Question 2: Let us assume none of the above attributes are calculated, nor non-atomic. Is R in 3NF? If not, then normalize it in 3NF (give the corresponding schema)

Solution:

# Normal forms

- Exercise 3:
  - Given the relation R (A, B, C, D, E) and the following FD:
    1.  $A \rightarrow B$
    2.  $BC \rightarrow E$
    3.  $DE \rightarrow A$

Question 3: Are the relations which you obtain in BCNF? If not, then normalize them in BCNF (multiple solutions are possible).

Solution:

-

# Normal forms

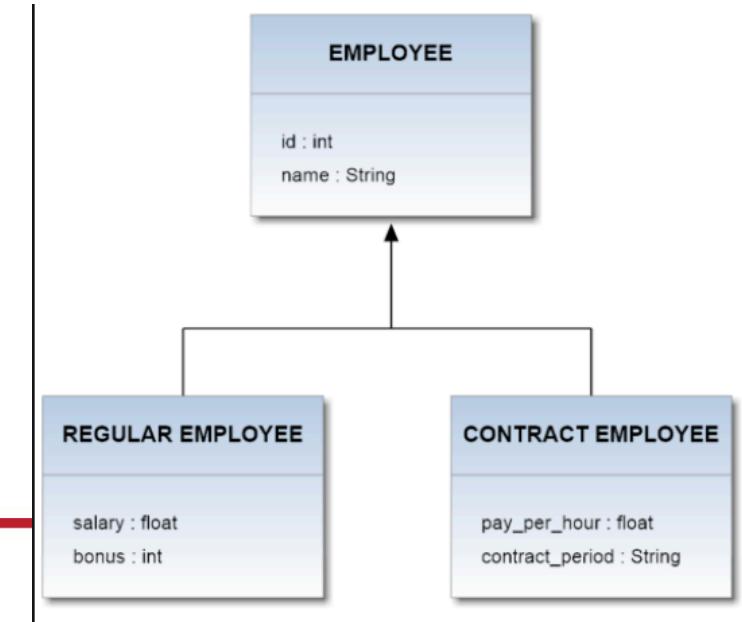
- Exercise 3:
  - Given the relation R (A, B, C, D, E) and the following FD:
    1.  $A \rightarrow B$
    2.  $BC \rightarrow E$
    3.  $DE \rightarrow A$

Question 4: Is there a loss of dependencies after you replied to Question 3? If yes, then which are the functional dependencies that are lost?

Solution:

# Some basic rules for a good database design...

- There are several rules for designing good relational databases, among which
  - No calculated attribute
    - *E.g.*: Total price, if it can be deduced from unit\_price and qtity
  - Normalized scheme
    - Decompose using the above algorithm until you reach (at least) 3NF (maybe even BCNF, depending on your needs)
    - « Merge » the entities which are semantically close, and share the same attributes (inheritance)
      - *E.g.*: movie\_director, movie\_producer, actor, viewer



# Exercises

- Do the exercise list number 5
  - Assignment due: Sunday, 9th of May, 23h59.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for  
your attention!

