

These slides are for use with

# Database Systems

## Concepts, Languages and Architectures

**Paolo Atzeni • Stefano Ceri • Stefano Paraboschi • Riccardo Torlone**  
© McGraw-Hill 1999

**To view** these slides on-screen or with a projector use the arrow keys to move to the next or previous slide. The return or enter key will also take you to the next slide. Note you can press the 'escape' key to reveal the menu bar and then use the standard Acrobat controls — including the magnifying glass to zoom in on details.

**To print** these slides on acetates for projection use the escape key to reveal the menu and choose 'print' from the 'file' menu. If the slides are too large for your printer then select 'shrink to fit' in the print dialogue box.

**Press the 'return' or 'enter' key to continue . . .**

# Chapter 8

## Normalization

# Database Systems

Concepts,  
Languages  
and  
Architectures

Paolo Atzeni,  
Stefano Ceri,  
Stefano Paraboschi  
and Riccardo Torlone

Mc  
Graw  
Hill

Click here  
for help ?

# Normal form and normalization

- A **normal form** is a property of a relational database.
- When a relation is *non-normalized* (that is, does not satisfy a normal form), then it presents redundancies and produces undesirable behavior during update operations.
- This principle can be used to carry out quality analysis and constitutes a useful tool for database design.
- **Normalization** is a procedure that allows the non-normalized schemas to be transformed into new schemas for which the satisfaction of a normal form is guaranteed.

1 NF  
3 NF  
BCNF

↓  
STORE STREET  
(LESS REDUNDANCY ACCURACY)

## Example of a relation with anomalies

Employee	Salary	Project	Budget	Function
Brown	20	Mars	2	technician
Green	35	Jupiter	15	designer
Green	35	Venus	15	designer
Hoskins	55	Venus	15	manager
Hoskins	55	Jupiter	15	consultant
Hoskins	55	Mars	2	consultant
Moore	48	Mars	2	manager
Moore	48	Venus	15	designer
Kemp	48	Venus	15	designer
Kemp	48	Jupiter	15	manager

The key is made up of the attributes Employee and Project

## Anomalies in the example relation

- The value of the salary of each employee is repeated in all the tuples relating to it: therefore there is a **redundancy**.
- If the salary of an employee changes, we have to modify the value in all the corresponding tuples. This problem is known as the **update anomaly**.
- If an employee stops working on all the projects but does not leave the company, all the corresponding tuples are deleted and so, even the basic information, name and salary is lost. This problem is known as the **deletion anomaly**.
- If we have information on a new employee, we cannot insert it until the employee is assigned to a project. This is known as the **insertion anomaly**.

## Why these undesirable phenomena?

- Intuitive explanation: we have used a single relation to represent items of information of different types.
- In particular, the following independent real-world concepts are represented in the relation:
  - employees with their salaries,
  - projects with their budgets,
  - participation of the employees in the projects with their functions.
- To systematically study the principles introduced informally, it is necessary to use a specific notion: the functional dependency.

THE TABLE  
DOES NOT  
HAVE A CLEAR  
SEMANTICS

→ AS WE SHALL SEE, THE  
SOLUTION IS PROVIDED  
BY TABLE DECOMPOSITION

CONFLICTING  
REQUIREMENTS

"DECORSE TO REDUCE REDUNDANCY"  
"TOO MANY TABLES REDUCE QUERY  
EFFICIENCY DUE TO JOIN OPERATIONS"  
© McGraw-Hill 1999

# Functional dependencies

RECORD  
INSTANCE

- Given a relation  $r$  on a schema  $R(X)$  and two non-empty subsets  $Y$  and  $Z$  of the attributes  $X$ , we say that there is a functional dependency on  $r$  between  $Y$  and  $Z$ , if, for each pair of tuples  $t_1$  and  $t_2$  of  $r$  having the same values on the attributes  $Y$ ,  $t_1$  and  $t_2$  also have the same values of the  $Z$  attributes.
- A functional dependency between the attributes  $Y$  and  $Z$  is indicated by the notation  $Y \rightarrow Z$ .

$$R(A_1, \dots, A_n) \quad Q, Z \subseteq X$$
$$X$$
$$Y \rightarrow Z \stackrel{\text{DEF.}}{=} \forall r \in R, \forall t_1, t_2 \in r, \\ \text{IF } t_1[x] = t_2[x] \text{ THEN } t_1[Q] = t_2[Q]$$

## Functional dependencies in the example schema

- Employee → Salary
  - the salary of each employee is unique and thus each time a certain employee appears in a tuple, the value of his or her salary always remains the same.
- Project → Budget
  - the budget of each project is unique and thus each time a certain project appears in a tuple, the value of its budget always remains the same.
- THESE CAN BOTH BE EASILY VIOLATED  
IN OUR TABLE -

## Non-trivial functional dependencies

- We then say that a functional dependency  $Y \rightarrow Z$  is **non-trivial** if

$Z \not\subseteq Q$

- Employee  $\rightarrow$  Salary is a non-trivial functional dependency
- Employee Project  $\rightarrow$  Project is a trivial functional dependency

## Anomalies and functional dependencies

- In our example, the two properties causing anomalies correspond exactly to attributes involved in functional dependencies:
  - the property ‘the salary of each employee is unique and depends only on the employee’ corresponds to the functional dependency  $\text{Employee} \rightarrow \text{Salary}$ ;
  - the property ‘the budget of each project is unique and depends only on the project’ corresponds to the functional dependency  $\text{Project} \rightarrow \text{Budget}$ .
- Moreover, the following property can be formalized by means of a functional dependency:
  - the property ‘in each project, each of the employees involved can carry out only one function’ corresponds to the functional dependency  $\text{Employee Project} \rightarrow \text{Function}$ .

However, there  
are no anomalies  
here ...

IT SEEMS THAT THE REASON FOR THE  
ANOMALY IS THAT WE HAVE ATTRIBUTES  
THAT “DO NOT DEPEND” ON THE WHOLE KEY,  
BUT JUST ON A PART OF IT ...

## Dependencies generating anomalies

- The first two dependencies generate undesirable redundancies and anomalies.
- The third dependency however never generates redundancies because, having Employee and Project as a key, the relation cannot contain two tuples with the same values of these attributes.
- The difference is that Employee Project is a key of the relation.

THUS, WE CANNOT HAVE 2 TUPLES WITH  
SAME EMPLOYEE AND PROJECT,  
BUT DIFFERENT FUNCTION

- $R = (A_1, \dots, A_m)$ ,  $X \subseteq \{A_1, \dots, A_m\}$  is a key of  $R$  if :
  - ①  $X \rightarrow \{A_1, \dots, A_m\}$  (i.e., if 2 tuples coincide on  $X$ , they must coincide on everything)
  - ②  $\forall Q \subsetneq X, Q \nrightarrow \{A_1, \dots, A_m\}$  (surjectivity requirement)

## Boyce–Codd Normal Form (BCNF)

- A relation  $R$  is in **Boyce–Codd normal form** if for every (non-trivial) functional dependency  $X \rightarrow Y$  defined on it,  $X$  contains a key  $K$  of  $r$ . That is,  $X$  is a superkey for  $R$  - *THOSE 2INES 3Y DOMAIN EXPERTS AND THOSE DERIVABLE FROM THEM*
- Anomalies and redundancies, as discussed above, do not appear in databases with relations in Boyce–Codd normal form, because the independent pieces of information are separate, one per relation.

*ALTHOUGH OTHER KINDS OF REDUNDANCIES MAY STILL BE PRESENT, E.G., THAT RELATED TO THE SO CALLED MULTIVALUED DEPENDENCIES*

COURSE	TEACHER	BOOK
PHYSICS	GREEN	INTRO TO RELATIVITY
PHYSICS	GREEN	OPTICS
TEACH	BROWN	ALGEBRA BASICS
HISTORY	DOE	ROMAN EMPIRE
HISTORY	DOE	MEDIEVAL TIMES

*PK = ALL ATTRIBUTES, SO IT IS IN BCNF; STILL, THERE IS REDUNDANCY*

## Decomposition into Boyce–Codd normal form

- Given a relation that does not satisfy Boyce–Codd normal form, we can often replace it with one or more normalized relations using a process called **normalization**.
- We can eliminate redundancies and anomalies for the example relation if we replace it with the three relations, obtained by projections on the sets of attributes corresponding to the three functional dependencies.  $\text{EMP} \rightarrow \text{SACRUE}$ ,  $\text{Proj} \rightarrow \text{BUDGET}$ ,  $\{\text{EMP}, \text{Proj}\} \rightarrow \text{BUDGET}$
- The keys of the relations we obtain are the left hand side of a functional dependency: the satisfaction of the Boyce–Codd normal form is therefore guaranteed.

$R_1(\underline{\text{EMP}}, \underline{\text{SACRUE}})$        $R_2(\underline{\text{Proj}}, \underline{\text{BUDGET}})$

$R_3(\underline{\text{EMP}}, \underline{\text{Proj}}, \underline{\text{BUDGET}})$

## Decomposition of the example relation

Employee	Salary
Brown	20
Green	35
Hoskins	55
Moore	48
Kemp	48

Project	Budget
Mars	2
Jupiter	15
Venus	15

HERE WE HAVE SIMPLY  
PROJECTED THE TUPLES  
OF THE ORIGINAL TABLE  
INTO THE THREE NEW  
TABLES -

Employee	Project	Function
Brown	Mars	technician
Green	Jupiter	designer
Green	Venus	designer
Hoskins	Venus	manager
Hoskins	Jupiter	consultant
Hoskins	Mars	consultant
Moore	Mars	manager
Moore	Venus	designer
Kemp	Venus	designer
Kemp	Jupiter	manager

## A relation to be decomposed

PK: {Employee, Branch}

Employee	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Venus	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

The relation **has** the functional dependencies:

- Employee → Branch
- Project → Branch

HOWEVER, THE RELATION  
IS NOT IN 3NF

LET US DECOMPOSE  
IT BASED ON  
THE FUNCTIONAL  
DEPENDENCIES

## A possible decomposition of the previous relation

Employee	Branch
Brown	Chicago
Green	Birmingham
Hoskins	Birmingham

Project	Branch
Mars	Chicago
Jupiter	Birmingham
Saturn	Birmingham
Venus	Birmingham

➡ IS THIS DECOMPOSITION OK?

## The join of the projections

Employee	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Venus	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham
Green	Saturn	Birmingham
Hoskins	Jupiter	Birmingham

→ SPURIOUS TUPLES

The result is different from the original relation: the information can not be reconstructed.

→ THE DECOMPOSITION IS CORRUPT

## Lossless decomposition

- The decomposition of a relation  $R$  on  $X_1$  and  $X_2$  is **lossless** if the join of the projections of  $R$  on  $X_1$  and  $X_2$  is equal to  $R$  itself (that is, not containing *spurious* tuples).
- It is clearly desirable, or rather an indispensable requirement, that a decomposition carried out for the purpose of normalization is lossless.

## A condition for the lossless decomposition

- Let  $R$  be a relation on  $X$  and let  $X_1$  and  $X_2$  be two subsets of  $X$  such that  $X_1 \cup X_2 = X$ . Furthermore, let  $X_0 = X_1 \cap X_2$ .
- If  $R$  satisfies the functional dependency  $X_0 \rightarrow X_1$  or the functional dependency  $X_0 \rightarrow X_2$ , then the decomposition of  $R$  on  $X_1$  and  $X_2$  is lossless.

IN THE PREVIOUS EXAMPLE :

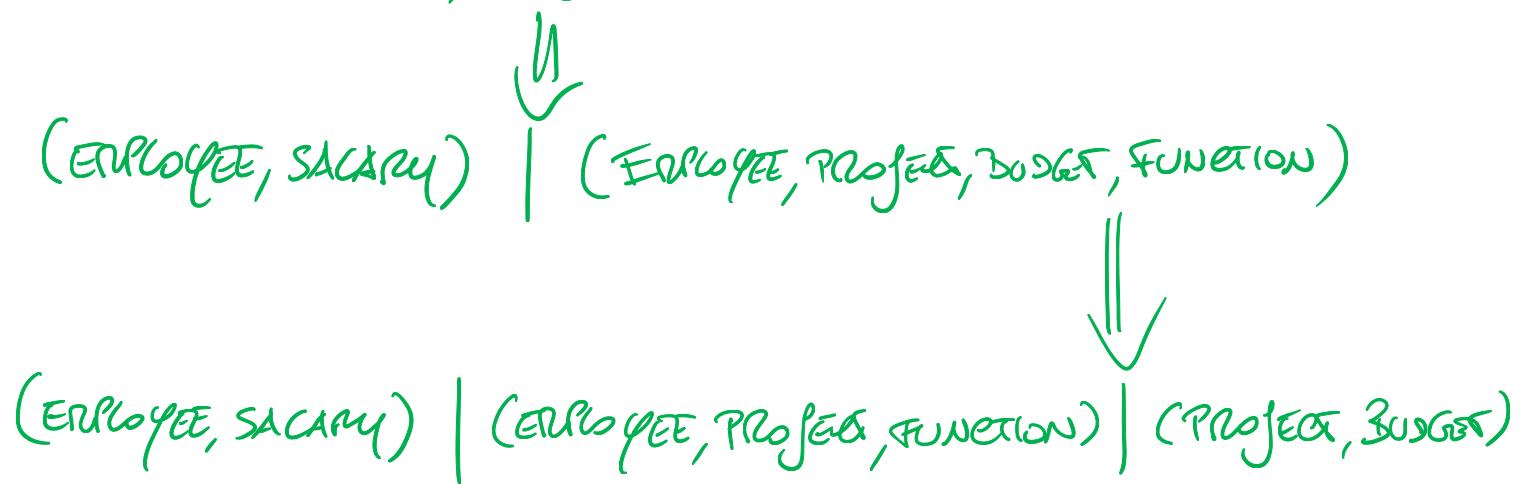
$R(\text{EMPLOYEE}, \text{PROJECT}, \text{BRANCH})$  WITH  $\{\text{EMPLOYEE} \rightarrow \text{Branch}$   
 $\text{PROJECT} \rightarrow \text{Branch}\}$

$R_1(\text{EMPLOYEE}, \text{BRANCH})$   $X_1$

$R_2(\text{PROJECT}, \text{BRANCH})$   $X_2$

$$\begin{aligned}X_1 \cup X_2 &= X \\X_1 \cap X_2 &= X_0 = \{\text{BRANCH}\} \\\text{BUT } \{\text{Branch} &\rightarrow X_1 \\&\text{Branch} \rightarrow X_2\}\end{aligned}$$

- THE RULE WE HAVE GIVEN WORKS FOR THE BINSAY DECOMPOSITION CASE
- HOWEVER, EVEN IF YOU HAVE A DECOMPOSITION WITH >2 TABLES, YOU CAN OBTAIN IT THROUGH A SERIES OF BINSAY DECOMPOSITIONS
- FOR INSTANCE : (EMPLOYEE, SALARY, PROJECT, BUDGET, FUNCTION)



## A lossless decomposition of the previous relation

$X_0 = \{\text{Employee}\}$

$X_0 \rightarrow \{\text{Employee}, \text{Branch}\}$

Employee	Branch
Brown	Chicago
Green	Birmingham
Hoskins	Birmingham

Employee	Project
Brown	Mars
Green	Jupiter
Green	Venus
Hoskins	Saturn
Hoskins	Venus

⇒ ARE WE HAPPY WITH THIS  
DECOMPOSITION NOW?

## Another problem with the new decomposition

- Assume we wish to insert a new tuple that specifies the participation of the employee named Armstrong, who works in Birmingham, on the Mars project.
- In the original relation this update would be immediately identified as illegal, because it would cause a violation of the Project → Branch dependency.
- On the decomposed relations however, it is not possible to reveal any violation of dependency since the two attributes Project and Branch have been separated: one into one relation and one into the other.

## Preservation of dependencies

- A decomposition **preserves the dependencies** if each of the functional dependencies of the original schema involves attributes that appear all together in one of the decomposed schemas.
- It is clearly desirable that a decomposition preserves the dependencies since, in this way, it is possible to ensure, on the decomposed schema, the satisfaction of the same constraints as the original schema.

## Qualities of decompositions

- Decompositions should always satisfy the properties of lossless decomposition and dependency preservation:
  - Lossless decomposition ensures that the information in the original relation can be accurately reconstructed based on the information represented in the decomposed relations.
  - Dependency preservation ensures that the decomposed relations have the same capacity to represent the integrity constraints as the original relations and thus to reveal illegal updates.

## A relation not satisfying the BCNF

Manager	Project	Branch
Brown	Mars	Chicago
Green	Jupiter	Birmingham
Green	Mars	Birmingham
Hoskins	Saturn	Birmingham
Hoskins	Venus	Birmingham

Assume that the following dependencies are defined:

- Manager → Branch: each manager works at a particular branch;
- Project Branch → Manager: each project has more managers who are responsible for it, but in different branches, and each manager can be responsible for more than one project; however, for each branch, a project has only one manager responsible for it.

THUS, KEYS ARE:

$K_1 : \{PROJECT, BRANCH\}$

$K_2 : \{PROJECT, MANAGER\}$

## A problematic decomposition

- The relation is not in Boyce–Codd normal form because the left hand side of the first dependency is not a superkey.
- At the same time, no good decomposition of this relation is possible: the dependency Project Branch → Manager involves all the attributes and thus no decomposition is able to preserve it.
- We can therefore state that sometimes, Boyce–Codd normal form cannot be achieved.

## A new normal form

SO WE CONSIDER A  
LESS STRICT NORMAL  
FORM, THAT CAN NEVERTHELESS  
ALWAYS BE ACHIEVED

- A relation  $r$  is in **third normal form** if, for each (non-trivial) functional dependency  $X \rightarrow Y$  defined on it, at least one of the following is verified:
  - $X$  contains a key  $K$  of  $r$ ; (*condition of 3NF*)
  - each attribute in  $Y$  is contained in at least one key of  $r$ .

## BCNF and third normal form

- The previous schema does not satisfy the Boyce–Codd normal form, but it satisfies the third normal form:
  - The Project Branch → Manager dependency has as its left hand side a key for the relation, while Manager → Branch has a unique attribute for the right hand side, which is part of the Project Branch key.
- The third normal form is less restrictive than the Boyce–Codd normal form and for this reason does not offer the same guarantees of quality for a relation; it has the advantage however, of always being achievable.

## Decomposition into third normal form

- Decomposition into third normal form can proceed as suggested for the Boyce–Codd normal form:
  - a relation that does not satisfy the third normal form is decomposed into relations obtained by projections on the attributes corresponding to the functional dependencies.
- The only condition to guarantee in this process is of always maintaining a relation that contains a key to the original relation.

$R(X)$ ,  $FD = \{X_1 \rightarrow Q_1, X_2 \rightarrow Q_2, X_3 \rightarrow Q_3\}$

$R_1(X_1, Q_1)$      $R_2(X_2, Q_2)$      $R_3(X_3, Q_3)$

- THEN, IF SOME OF THESE RELATIONS IS INCLUDED IN ANOTHER ONE, JUST REMOVE IT
- IF NONE OF THE DECOMPOSED RELATIONS INCLUDE A KEY OF THE ORIGINAL ONE, ADD A RELATION CONTAINING A KEY OF THE ORIGINAL RELATION

RESOURCING TEAM  
A BETTER ANALYSIS  
OF STAKEHOLDER  
REQUIREMENTS

## A restructuring of the previous relation

Manager	Project	Branch	Division
Brown	Mars	Chicago	1
Green	Jupiter	Birmingham	1
Green	Mars	Birmingham	1
Hoskins	Saturn	Birmingham	2
Hoskins	Venus	Birmingham	2

Functional dependencies:

- Manager → Branch Division: each manager works at one branch and manages one division;
- Branch Division → Manager: for each branch and division there is a single manager;
- Project Branch → Division: for each branch, a project is allocated to a single division and has a sole manager responsible.

THUS, KEYS :

$K_1: \{MANAGER, PROJECT\}$

$K_2: \{BRANCH, DIVISION, PROJECT\}$

• WHAT NORMAL FORM DO WE HAVE HERE?  
 © McGraw-Hill 1999

## A good decomposition of the restructured schema

$\{ \text{MANAGER} \rightarrow \{ \text{BRANCH}, \text{DIVISION} \}$   
 $\{ \text{BRANCH}, \text{DIVISION} \} \rightarrow \text{MANAGER}$

$\{ \text{Project}, \text{Branch} \} \rightarrow \text{Division}$

Manager	Branch	Division
Brown	Chicago	1
Green	Birmingham	1
Hoskins	Birmingham	2

$X_0 = \{ \text{BRANCH}, \text{DIVISION} \}$

Project	Branch	Division
Mars	Chicago	1
Jupiter	Birmingham	1
Mars	Birmingham	1
Saturn	Birmingham	2
Venus	Birmingham	2

- The decomposition is lossless and the dependencies are preserved.
- This example shows that often the difficulty of achieving Boyce–Codd normal form could be due to an insufficiently accurate analysis of the application.