CSCI235 Database Systems

Functional Dependencies

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Functional dependencies

Outline

Functional dependency ? What is it ?

Let $R = (A_1, \ldots, A_n)$ be a relational schema (a header of relational table) and let X, Y be the nonempty subsets of R

We say that a functional dependency X → Y is valid in a relational schema R if

for any contents of a relational table R, it is not possible that R has two rows that agree in the components for all attributes in a set X yet

disagree on one or more component for the attributes in a set Y

Examples

- A warehouse is located at exactly one address: warehouse → address
- An address is related to exactly one warehouse: address → warehouse
- At a warehouse, the parts of the same sort have only one total quantity: warehouse,part → quantity
- A car has one owner: registration → driving license
- A student has one first name and one last name and one date of birth: student-number → first-name, last-name-date-of-birth

Functional dependency ? What is it ?

More examples

- An employee belongs to one department: employee-number → department-name
- A manager manages one department: manager-number → department-name
- An employee has one manager: employee-number → manager-number
- A student enrols a subject one time: student-number,subject-code → enrolment-date
- An employee is located in one building in one office: employee-number → building-number, office-number
- An office in a building hosts one employee: building-number, office-number → employee-number
- An office in a building at a campus hosts one employee: campus-name,building-number, office-number → employee-number
- A department has one manager: department-name → manager-number
- A department is located in one building: department-name → building-number
- A department has one manager and it is located in one building: department-name → manager-number,building-number

Functional dependency ? What is it ?

How to discover the functional dependencies in a relational table ?

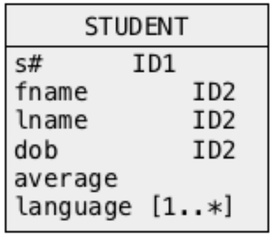
- Is it possible to discover the functional dependencies in a relational schema (a header of relational table) `R(A, B, C, D, E)` ?
- Of course it is impossible to do it because we do not know the semantics (the meanings) of the names: `R, A, B, C, D, E`
- To discover the functional dependencies in a relational table we must use the semantics of a relational table name and the names of attributes
- For example consider a relational schema (a header of relational table) `TRIP(rego#, licence#, tdate)` of a relational table that contains information about the trips made by the drivers (`licence#`) who used the trucks (`rego#`) on a given day (`tdate`)
- Can a truck be used only one time ? If yes then rego# → tdate
- Can a driver make only one trip ? If yes then licence# → tdate
- Can a driver use more than one truck ? If no then licence# → rego#
- Can a truck be used by more than one driver ? If no then rego# → licence#
- And so on ...

Functional dependencies

Outline

Functional dependencies versus classes of objects

A class of objects `STUDENT`



validates (satisfies) the following functional dependencies:

s# → fname
s# → lname
s# → dob
s# → average
fname, lname, dob → s#
fname, lname, dob → average

Functional dependencies versus classes of objects

The functional dependencies:
s# → fname
s# → lname
s# → dob
s# → average
are equivalent to a functional dependency
s# → fname, lname, dob, average

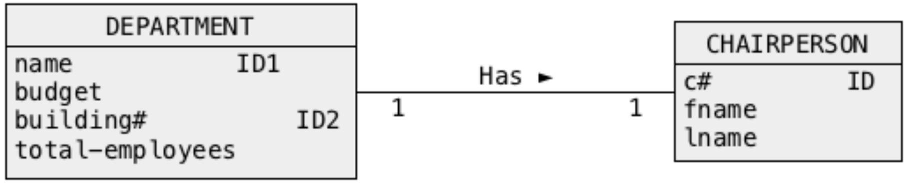
The functional dependencies
fname, lname, dob → s#
fname, lname, dob → average
are equivalent to a functional dependency
fname, lname, dob → s#, average

Functional dependencies

Outline

Functional dependencies versus associations

The classes of objects `DEPARTMENT` and `CHAIRPERSON` and association `Has`



validate (satisfy) the following functional dependencies:

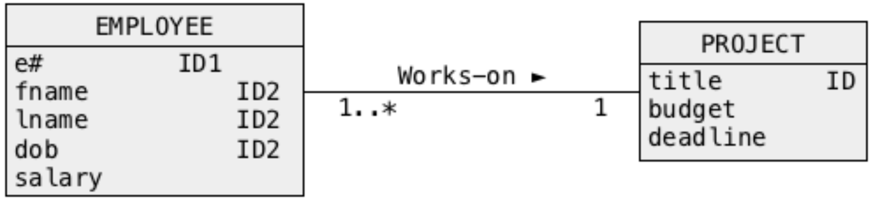
name → budget, building#, total-employees
building# → name, budget, total-employees

c# → fname, lname

name → c#, fname, lname
building# → c#, fname, lname
c# → name, building#, budget, total-employees

Functional dependencies versus associations

The classes of objects `EMPLOYEE` and `PROJECT` and association `Works-on`



validate (satisfy) the following functional dependencies:

e# → fname, lname, dob, salary
fname, lname, dob → e#, salary

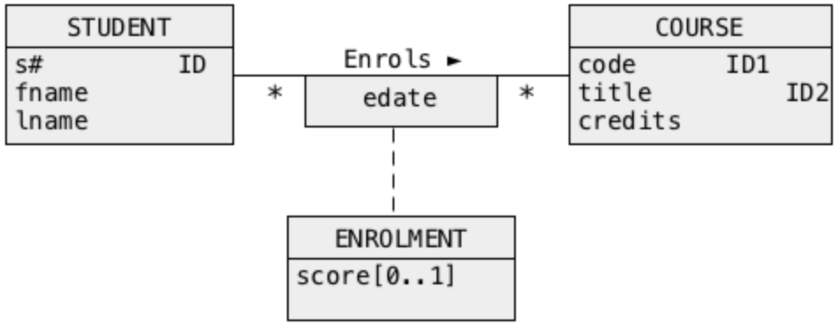
title → budget, deadline

e# → title, budget, deadline

fname, lname, dob → title, budget, deadline

Functional dependencies versus associations

The classes of objects `STUDENT` and `COURSE` and association `Enrols`



validate (satisfy) the following functional dependencies:

s# → fname, lname

code → title, credits
title → code, credits

s#, code, edate → score
s#, title, edate → score

Functional dependencies

Outline

Derivations of functional dependencies

Consider a relational schema (a header of relational table)
`EMPLOYEE(e#, ename, department, address, chairperson)`

If e# → ename and e# → department then e# → ename, department

If e# → department and department → address then e# → address

If e# → department and department → chairperson then e# → chairperson

If e# → department then e#, ename → department

If e#, ename → department then e#, ename, address → department

It is always true that e# → e#

Functional dependency e# → e# is called as a trivial functional dependency

It is always true that e#, ename → e#

A functional dependency e#, ename → e# is also called as a trivial functional dependency

In HTML view press 'p' to see the lecture notes

Derivations of functional dependencies

A trivial functional dependency is a functional dependency that is always true no matter what its left and right hand sides are

For example,
e# → e#,
department → department
e#, ename → e#,
e#, ename, department → e#, department,
and so on

Derivations of functional dependencies

Consider a relational schema `R(A, B, C)`

It is always true that $A \rightarrow A$

It is always true that $A, B \rightarrow A$

It is always true that $A, B, C \rightarrow A$

If $A \rightarrow B$ then $A, C \rightarrow B$

If $A \rightarrow B, C$ then $A \rightarrow B$ and $A \rightarrow C$

If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

Functional dependencies

Outline

Armstrong axioms

Let $R = (A_1, \ldots, A_n)$ be a relational schema (a header of relational table) and
let X, Y, Z be the nonempty subsets of $\{A_1, \ldots, A_n\}$

(i) If $Y \subseteq X$ then $X \rightarrow Y$ (reflexivity axiom)

(ii) If $X \rightarrow Y$ then $X, Z \rightarrow Y, Z$ (augmentation axiom)

(iii) If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$ (transitivity axiom)

The axioms (i),(ii), and (iii) form a minimal and complete set of axioms

Functional dependencies

Outline

Other inference rules

Let $R = (A_1, ..., A_n)$ be a relational schema (a header of relational table) and
let X, Y, Z be the nonempty subsets of $\{A_1, ..., A_n\}$

If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow Y, Z$ (union rule)

If $X \rightarrow Y$ and $W, Y \rightarrow Z$ then $W, X \rightarrow Z$ (pseudotransitivity rule)

If $X \rightarrow Y$ and $Z \subseteq Y$ then $X \rightarrow Z$ (decomposition rule or reduce right hand side rule)

If $X \rightarrow Y$ then $X, Z \rightarrow Y$ (extend left hand side rule)

Functional dependencies

Outline

Using inference rules

Let R = (A, B, C) be a relational schema

Given set of functional dependencies F = {A → B, B → C} valid in R

Is it true that A → C ?

If A → B and B → C then application of transitivity axiom provides A → C

Using inference rules

Let R = (A, B, C) be a relational schema

Given set of functional dependencies F = {A → B, C} valid in R

Is it true that A → B and A → C ?

Reflexivity axiom provides B, C → C

If A → B, C and B, C → C then transitivity axiom provides A → C

Reflexivity axiom provides B, C → B

If A → B, C and B, C → B then transitivity axiom provides A → B

Using inference rules

Let R = (A, B, C) be a relational schema

Given set of functional dependencies F = {A → B, A → C} valid in R

Is it true that A → B, C ?

If A → B then augmentation axiom provides A → A, B

If A → C then augmentation axiom provides A, B → B, C

If A → A, B and A, B → B, C then transitivity axiom provides A → B, C

Using inference rules

Let R = (A, B, C) be a relational schema

Given set of functional dependencies F = {A → B} valid in R

Is it true that A, C → B ?

Reflexivity axiom provides A, C → A

If A, C → A and A → B then transitivity axiom provides A, C → B

Using inference rules

A relational schema STUDENT(s#, fname,lname, dob, average) validates (satisfies) the following functional dependencies:
s# → fname
s# → lname
s# → dob
s# → average
fname, lname, dob → s#
fname, lname, dob → average

We proved that if A → B and A → C then A → B, C

Hence,
s# → fname, lname, dob, average and ...

fname, lname, dob → s#, average

Note, that both functional dependencies cover entire relational schema and no other functional dependencies that do not cover entire relational schema validate in the schema e.g. fname → s#

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 14.4 Functional Dependencies, Chapter 15.1 More on Functional Dependencies, Pearson Education Ltd, 2015