Designing a Relational Database
Functional dependencies

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.

Schema

Definition

A **relation schema** R is a set of attributes $\{A_1, A_2, ..., A_n\}$

Notation:

- $R = A_1 A_2 ... A_n$
- first letters of the alphabet (A,B,C,...) denote single attributes
- last letters of the alphabet (X,Y,...) denote sets of attributes
- if X and Y are sets of attributes XY denotes the union of X and Y

Tuple

definition:

- given a relation $R = A_1\ A_2 \dots A_n$,
- a **tuple** t on R is a **function** that associates to each attribute A_i in R a value $t[A_i]$ in the corresponding domain $dom(A_i)$

	Name	Surname	PassedEx	Avg
t_1	Paolo	Rossi	2	26.5
t_2	Mario	Bianchi	10	28.7

t_1[Name]=Paolo
t_1[Surname]=Rossi
t_1[PassedEx]=2
t_1[Avg]=26.5

t_2[Name]=Mario
t_2[Surname]=Whites
t_2[PassedEx]=10
t_2[Avg]=28.7

Tuple

if X is a subset of R and t_1 and t_2 are two tuples on R:

t_1 and t_2 coincide on X ($t_1[X]=t_2[X]$)

if $\forall A \in X$: $t_1[A]=t_2[A]$

	Name	Surname	PassedEx	Avg
t_1	Paolo	Rossi	3	27
t_2	Mario	Rossi	5	27

t_1[Surname Avg] = t_2[Surname Avg]

t_1[Surname Name] != t_2[Surname Name]

Instance of a schema

definition:

- given a relation schema *R*,
- an **instance** of *R* is a set of tuples on *R*
- **ALL the "tables" we have seen so far in the examples are <u>instances</u> of some schema**

Functional dependency

definition:

- given a relation schema R,
- a **functional dependence** on R is an ordered pair of non-empty subsets $X, Y \subseteq R$

notation and terminology:

- $X \rightarrow Y$
- X functionally determines Y or
- Y is functionally dependent on X
- **X = left-hand side of the dependency, or <u>determinant</u>**
- **Y = right-hand side of the dependency, or <u>dependent</u>**

Functional dependency

definition:

given a schema R and a functional dependence X $\rightarrow$ Y defined on R,

an instance *r* of R **satisfies** the functional dependence $X \rightarrow Y$ if:

$$\forall\, t_1, t_2 \in r : t_1[X]=t_2[X] \Rightarrow t_1[Y]=t_2[Y]$$

note: obviously, if $t_1[X] \neq t_2[X]$ the dependence is satisfied whatever the values of $t_1[Y]$ and $t_2[Y]$

Functional dependencies

Note:

implies

$$\forall\, t_1, t_2 \in r : t_1[X]=t_2[X] \Rightarrow t_1[Y]=t_2[Y]$$

means that:

$\forall\, t_1, t_2 \in r :$ ***if*** $(t_1[X]=t_2[X]$ ***then*** $t_1[Y]=t_2[Y])$

it does not mean that:

$\cancel{\forall\, t_1, t_2 \in r : \textit{if } (t_1[X]!=t_2[X] \textit{ then } t_1[Y]!=t_2[Y])}$

Note

- functional dependencies do nothing more than express **constraints** on data

- e.g., if two tuples have the same tax code (so, they refer to the same person) they must also have the same date of birth!

Note

- in the schema representing the exams, **we do not have**
- Grade→Honors, as if t_1[Grade]=t_2[Grade]=27 then it must surely be t_1[Honors]=t_2[Honors]='No' but ...
- ... if t_1[Grade]=t_2[Grade]=30 and t_1[Honors]='Yes' this does not determines the value of t_2[Honors] (it could be 'Yes' or 'No', without compromising the correctness of the data)

- can we say that Grade⟶Honors?

- if t_1[Honors]=t_2[Honors]= 'Yes' then it must surely be t_1[Grade]=t_2[Grade]=30 but ...
- ... if t_1[Honors]=t_2[Honors]='No' and t_1[Grade]=27 this does not determine the value of t_2[Grade] (it can be any number between 18 and 30)

- again, **we cannot say that** Honors⟶ Grade

Note

- since certain properties apply **regardless of the specific attributes involved**, we will use an "abstract" notation and assume that the dependencies **have already** been defined

Example

R

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

satisfies the functional dependence AB $\rightarrow$ C

Example

R

A	B	C	D
a1	**b1**	**c1**	d1
a1	**b1**	**c2**	d2
a1	b2	c1	d3

does not satisfy the functional dependence $AB \rightarrow C$

Legal instance

- given a relation schema *R* and a set *F of* functional dependencies defined on it,
- an instance of *R* is **legal** if it satisfies **all** dependencies in *F*

Observation

F={A→B}

R

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

- **the instance satisfies** the functional dependency *A→B* (therefore, it is a **legal** instance)
- ...the functional dependence *A→C* is also satisfied ... but ... *A→C* **is not** in F, so it does not necessarily have to be satisfied!

Observation

F={A→B}

R	A	B	C	D
	a1	b1	c1	d1
	a1	b1	c2	d2
	a2	b2	c1	d3

this instance satisfies the functional dependency *A* →*B* (so, it is a **legal** instance)

but it **does not satisfy** the functional dependence *A*→*C*, *on the* other hand ... *A*→*C* is not satisfied in *F* so ... why should it be satisfied anyway?

Observation

F={A→B, B→C}

R

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

each legal instance (i.e., each instance satisfying both *A→B* and *B→C* **also** always **satisfies** the functional dependence *A→C*... can we consider it "*as if it was in F*"?

Conclusion

given a relation schema *R* and a set *F of* functional dependencies on R there are functional dependencies **that are not in *F***, but **are satisfied by every legal instance of *R***

A few examples

- Tax Code → First Name, Last Name
- must be met by every legal instance
- ...but then:
- Tax Code → Name and
- Tax Code → Surname

Closure of a set of functional dependencies

definition:

- given a relational schema *R* and a set *F of* functional dependencies defined on *R*
- the **closure of *F*** is the set of functional dependencies that are satisfied **by each legal instance** of *R*

Notation

- $\boldsymbol{F^+}$

F and F^+

- if *F is a* set of functional dependencies on *R* and *r* is an instance of *R* that satisfies **all** dependencies in *F*, we say that *r* is ***a legal instance*** of *R*

- the *closure of F,* denoted as F^+*,* is the set of functional dependencies that are satisfied **by each** legal instance of *R*

- trivially, we have that $F \subseteq F^+$

Key

definition:

given a relation schema *R* and a set *F of* functional dependencies defined on R:

a subset K of a relation schema R is a **key** of R if:

1) $K \rightarrow R \in F^+$
2) there is no proper subset $K' \subseteq K$ such that $K' \rightarrow R$

consider the schema:

Student = Matr, LastName, FirstName, BirthD

the student's matriculation number identifies the student

there cannot be two students with the matriculation number

an instance of Student cannot contain two tuples with the same matriculation number that correctly represents reality

$$\text{Matr} \rightarrow \text{Surname FirstName BirthD}$$

must be met by every legal instance

Matr is a key of Student

Primary key

- given a schema R and a set F of functional dependencies, there can be **multiple keys** of R
- in SQL, one of them will be chosen as **primary key** (it cannot be assigned a null value)

- example:

  Student=<u>Matr</u>,TaxC,Surname,Name,BirthD

Trivial functional dependencies

- given a 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 dependence $X \rightarrow Y$

R

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

X=ABC
Y=AB

X→Y is satisfied

Trivial functional dependencies

therefore, if $Y \subseteq X$ then $X \rightarrow Y \in F^+$

such a functional dependency is called **trivial**

Functional dependencies (properties)

- given a 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:

- if $t_1[X]=t_2[X]$ **then it must be** $t_1[Y]=t_2[Y]$
- obviously, if $\mathbf{A \rightarrow Y}$ and $\mathbf{t_1[A] \neq t_2[A]}$**, it cannot be** $t_1[Y]=t_2[Y]$
- and if $\mathbf{A \rightarrow Y}$ and $\mathbf{t_1[A]=t_2[A]}$**, it will be** $t1[Y]=t2[Y]$

R

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

$A \rightarrow BC \in F+$

$A \rightarrow B \in F+$

$A \rightarrow C \in F+$

Problem

- we will see later, talking about the ***3rd normal form (3NF)*** that the set F^+ is very important

- but... how can we calculate F^+?