Database Systems

Relational Design - Functional Dependencies

Prof. Dr. Agnès Voisard   Muhammed-Ugur Karagülle

Institute of Computer Science, Databases and Information Systems Group

Fraunhofer FOKUS

2025

Notes

Notes

Outline

Designing a database:

- Common sense, intuition of DB designer
- Mapping ER schema onto relational schema

No final measure of why one grouping of attributes was better than another

Theory to attempt to design good relational schemas 2 levels:

- logical level
- storage level

Main tool for measuring the approprietness of attribute grouping into relation schemas: **functional dependencies**

Constraints on the set of relations
Generalizes the notion of superkey

Example

EMPLOYEE(SSN, Name, Address)
SECRETARY(SSN, Salary)

"In relation **EMPLOYEE** if we know a SSN we know the name"

Attribute SSN **determines** attribute **Name**,
or Name is **functionally determined** by SSN

Functional Dependencies (cont'd)

Notation:

SSN $\rightarrow$ **Name**

In relation SECRETARY: **SSN** $\rightarrow$ **Salary**

BUT: if **we know the name** of an employee (or the **salary** of a secretary)

we **do not know** his/her **SSN**

Name $\not\rightarrow$ SSN

Salary $\not\rightarrow$ SSN

Functional Dependencies - Example

Example

`COURSE(CourseName,Teacher,Books)`

CourseName	Teacher	Books
Data Structures	Müller	Knuth
Databases	Müller	Ullman
Databases	Müller	Date
Compilers	Meier	Aho et al.

A course has one teacher
$\Leftrightarrow$ CourseName $\rightarrow$ Teacher
$\Leftrightarrow$ 2 tuples having the same value for CourseName have the same value for Teacher

- Tuple (`Databases, Meier, Ullman`)?

`EMPLOYEE(SSN,Name,Address)`

SSN $\rightarrow$ Name
SSN $\rightarrow$ Address

If we know SSN we know the other attributes
$\Leftrightarrow$ 2 tuples having the same SSN are identical
$\Leftrightarrow$ SSN identifies a tuple
SSN is a **key**

If we know SSN and Name we know the address
SSN, Name $\rightarrow$ Address
But Name is non necessary
(SSN, Name) is a **superkey**

R (U): relation scheme

Subset K of U is a **superkey** of R

if in any relation r of schema R(U)

for all pairs t_1 and t_2 of tuples in r such that $t_1 \neq t_2$:

$$t_1[K] \neq t_2[K]$$

(no 2 tuples in any r(R) may have the same value on attribute set K)

Let $X \subseteq U$ and $Y \subseteq U$

Functional dependency (FD):

$X \rightarrow Y$

holds on R

if in any relation r(R),

for all pairs t_1 and t_2 of tuples in r such that $t_1[X] = t_2[X]$

it is also the case that $t_1[Y] = t_2[Y]$

K is a **superkey** of R if $K \rightarrow U$

($K \rightarrow U - K$, $K \rightarrow K$)

i.e.,
K is a superkey if whenever $t_1[K] = t_2[K]$,
it is also the case that $t_1[U] = t_2[U]$ (i.e., $t_1 = t_2$)

K is a **key** if:

- K is a superkey: $K \rightarrow U$
- It does not exist $Y \subset K$ such that $Y \rightarrow U$ (all the attributes of K are <u>necessary</u> to determine all the attributes of the schema)

Use of Functional Dependencies

Specify constraints on a set of relations

Be concerned *only* with relations that satisfy a given set of functional dependencies F: **legal relations**

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b3	c2	d3
a3	b3	c2	d4

A$\rightarrow$C

There are 2 tuples with value a_1 for A; they have the same value for C

After looking at a_1,a_2, a_3 : A$\rightarrow$C satisfied

Is C$\rightarrow$ A satisfied?

Trivial FD

Other dependencies: AB $\rightarrow$ D

A $\rightarrow$ A satisfied by all relations involving attribute A

Trivial FD

Idem for AB $\rightarrow$ A

In general FD of the form X$\rightarrow$Y trivial if Y$\subseteq$X

Functional Dependencies - Example

When a relational DB is designed, first look at the FDs that must always hold Example: Bank database

Example

BRANCH(BranchName, Assets, BranchCity)

FD_{BRANCH} = BranchName → BranchCity, BranchName → Assets

CUSTOMER(CName, Street, City)

$FD_{CUSTOMER}$ = CName → City, CName → Street

DEPOSIT(BranchName, AccountNum, CName, Balance)

$FD_{DEPOSIT}$ = AccountNum → BranchName, AccountNum → Balance

BORROW(BranchName, LoanNum, CName, Amount)

FD_{BORROW} = LoanNum → Amount, LoanNum → BranchName

Closure of a Set of FDs

Not enough to consider a given set of FDs: consider **all** FDs that hold

Set F of functional dependencies, other dependencies hold: logically implied by F ($F \models fd$)

Example

R = (A, B, C, G, H, I)

F: { $A \rightarrow B$, $A \rightarrow C$, $CG \rightarrow H$, $CG \rightarrow I$, $B \rightarrow H$ }

$A \rightarrow H$ is logically implied ($F \models A \rightarrow H$)

Suppose that $t_1[A] = t_2[A]$

Since A→B: $t_1[B] = t_2[B]$
Since B→H: $t_1[H] = t_2[H]$

Then if we have $t_1[A] = t_2[A]$ it must be that $t_1[H] = t_2[H]$
Definition of $A \rightarrow H$

Closure of F:
Set of functional dependencies logically implied by F
Denoted by F^+
$F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$

Armstrong's Axioms:

- **Reflexivity** rule
  If X⊆Y then $Y \to X$
- **Augmentation** rule
  If X→Y and Z is a set of attributes then XZ $\to$ YZ
- **Transitivity** rule
  If X→Y and $Y \to Z$ then X→ Z

Complete rules: Allows to generate all of F^+

Additional rules:

- **Union** rule
  If X→Y and X→ Z then X→ YZ
- **Decomposition** rule
  If X→ YZ then X→Y and X→ Z
- **Pseudotransitivity** rule
  If X→Y and ZY → T then XZ → T

Closure of a Set of FDs (cont'd)

Example

R = (A, B, C, G, H, I)

F: $\{A \rightarrow B,\ A \rightarrow C,\ CG \rightarrow H,\ CG \rightarrow I,\ B \rightarrow H\}$

Some members of F^+:

- $A \rightarrow H$
- $CG \rightarrow HI$
- $AG \rightarrow I$
- $\ldots$

F and F set of dependencies
F and F *equivalent* if $F^+ = G^+$ (F "covers" F and F "covers" F)

Set of dependencies **minimal** if:

1. Every right side of a dependency in F is a **single attribute** (apply decomposition)
2. For no X$\rightarrow$ A in F is the set $F - \{X \rightarrow A\}$ equivalent to F (**all dependencies are useful**)
3. For no X$\rightarrow$ A in F and subset Z of X is $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ equivalent to F (**no attribute in any left side is redundant**)

Every set of dependencies is equivalent to a set F' that is minimal

To find out whether a set X is a superkey:

Compute the set of attributes functionally determined by X

Closure of X under F, denoted X^+

Algorithm:

result := *X*
while *result changes* ***do***
   for each *FD* $Y \rightarrow Z$ ***in*** *F* ***do***
      if $Y \subseteq$ *result* ***then*** *result* := *result* $\cup$ *Z*

Example: $(AG)^+$

New Definition of a Superkey

X is a superkey if for each $A \in U$

$F \models X \rightarrow A$
($X \rightarrow A$ can be inferred from F)

or if $X \rightarrow A \in F^+$

To compute a superkey it is not necessary to know F^+:

Find $X \subset U$ such that
$X^+ = U$

Key if there does not exist $Y \subset X$ such that $Y^+ = U$
=> Find the "smallest" X such that $X^+ = U$
Each one is a key

Summary

- Functional dependencies (FDs)
- Key, superkey
- Closure of a set of FDs (Armstrong's axioms)
- Minimal cover of a set of FDs
- Closure of attribute set

Questions?

What will come next?

Example

$F = \{AB \rightarrow C,\ C \rightarrow A,\ BC \rightarrow D,\ ACD \rightarrow B,\ D \rightarrow EG,\ BE \rightarrow C,\ CE \rightarrow AG,\ CG \rightarrow BD\}$

Split right-hand sides:

F $=\{AB \rightarrow C,\ C \rightarrow A,\ BC \rightarrow D,\ ACD \rightarrow B,\ D \rightarrow E,\ D \rightarrow G,\ BE \rightarrow C,\ CE \rightarrow A,\ CE \rightarrow G,\ CG \rightarrow B,\ CG \rightarrow D\}$

- CE $\rightarrow$ A is redundant (since C $\rightarrow$ A)
- CG $\rightarrow$ B is redundant (from CG $\rightarrow$ D, C $\rightarrow$ A, ACD $\rightarrow$ B)

Minimal cover:

F' $=\{AB \rightarrow C,\ C \rightarrow A,\ BC \rightarrow D,\ ACD \rightarrow B,\ D \rightarrow E,\ D \rightarrow G,\ BE \rightarrow C,\ CE \rightarrow G,\ CG \rightarrow D\}$