

FD examples

Address (street_address, city, state, zip)

4

_____ redefined using FD's

5

Reasoning with FD's

6

Given a relation R and a set of FD's \mathcal{F}

- ❖ Does another FD follow from \mathcal{F} ?
 - Are some of the FD's in \mathcal{F} redundant (i.e., they follow from the others)?
- ❖ Is K a key of R ?
 - What are all the keys of R ?

Attribute closure 7

❖ Given R , a set of FD's \mathcal{F} that hold in R , and a set of attributes Z in R :

The closure of Z (denoted Z^+) with respect to \mathcal{F} is the set of all attributes functionally determined by Z

❖ Algorithm for computing the closure

- Start with closure = Z
- If $X \rightarrow Y$ is in \mathcal{F} and X is already in the closure, then also add Y to the closure
- Repeat until no more attributes can be added

A more complex example 8

StudentGrade (SID, name, email, CID, grade)

❖ Not a good design, and we will see why later

Example of computing closure 9

❖ \mathcal{F} includes:

❖ $\{ CID, email \}^+ = ?$

Using attribute closure

10

Given a relation R and set of FD's \mathcal{F}

- ❖ Does another FD $X \rightarrow Y$ follow from \mathcal{F} ?
 - Compute X^+ with respect to \mathcal{F}
 - If $Y \subseteq X^+$, then $X \rightarrow Y$ follow from \mathcal{F}
- ❖ Is K a key of R ?
 - Compute K^+ with respect to \mathcal{F}
 - If K^+ contains all the attributes of R , K is a super key
 - Still need to verify that K is *minimal* (how?)

Rules of FD's

11

- ❖ Armstrong's axioms
 - Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$
 - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
 - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- ❖ Rules derived from axioms
 - Splitting: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
 - Combining: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

Using rules of FD's

12

Given a relation R and set of FD's \mathcal{F}

- ❖ Does another FD $X \rightarrow Y$ follow from \mathcal{F} ?
 - Use the rules to come up with a proof
 - Example:
 - \mathcal{F} includes:
 - $SID \rightarrow name, email; email \rightarrow SID; SID, CID \rightarrow grade$
 - $CID, email \rightarrow grade?$
 - $email \rightarrow SID$ (given in \mathcal{F})
 - $CID, email \rightarrow CID, SID$ (augmentation)
 - $SID, CID \rightarrow grade$ (given in \mathcal{F})
 - $CID, email \rightarrow grade$ (transitivity)

Non-key FD's

13

- ❖ Consider a non-trivial FD $X \rightarrow Y$ where X is not a super key
 - Since X is not a super key, there are some attributes (say Z) that are not functionally determined by X

X	Y	Z
a	b	$c1$
a	b	$c2$
...

Example of redundancy

14

- ❖ *StudentGrade* (SID , $name$, $email$, CID , $grade$)
- ❖ $SID \rightarrow name, email$

SID	$name$	$email$	CID	$grade$
142	Bart	bart@fox.com	CPS196	B-
142	Bart	bart@fox.com	CPS114	B
123	Milhouse	milhouse@fox.com	CPS196	B+
857	Lisa	lisa@fox.com	CPS196	A+
857	Lisa	lisa@fox.com	CPS130	A+
456	Ralph	ralph@fox.com	CPS114	C
...

Decomposition

15

SID	$name$	$email$	CID	$grade$
...

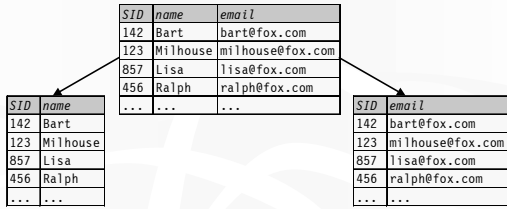
SID	$name$	$email$
142	Bart	bart@fox.com
123	Milhouse	milhouse@fox.com
857	Lisa	lisa@fox.com
456	Ralph	ralph@fox.com
...

SID	CID	$grade$
142	CPS196	B-
142	CPS114	B
123	CPS196	B+
857	CPS196	A+
857	CPS130	A+
456	CPS114	C
...

- ❖ Eliminates redundancy
- ❖ To get back to the original relation:

Unnecessary decomposition

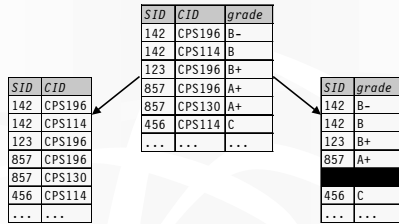
16



- ❖ Fine: join returns the original relation
- ❖ Unnecessary: no redundancy is removed, and now *SID* is stored twice!

Bad decomposition

17



- ❖ Association between *CID* and *grade* is lost
- ❖ Join returns more rows than the original relation

Lossless join decomposition

18

- ❖ Decompose relation R into relations S and T
 - $attrs(R) = attrs(S) \cup attrs(T)$
 - $S = \pi_{attrs(S)}(R)$
 - $T = \pi_{attrs(T)}(R)$
- ❖ The decomposition is a lossless join decomposition if, given constraints such as FD's, we can guarantee that $R = S \bowtie T$
- ❖ Any decomposition has $R \subseteq S \bowtie T$ (why?)
 - A lossy decomposition is one with $R \subset S \bowtie T$

Loss? But I got more rows!

19

- ❖ “Loss” refers not to the loss of tuples, but to the loss of information
 - Or, the ability to distinguish different original relations

<i>SID</i>	<i>CID</i>
142	CPS196
142	CPS114
123	CPS196
857	CPS196
857	CPS130
456	CPS114
...	...

<i>SID</i>	<i>CID</i>	<i>grade</i>
142	CPS196	B-
142	CPS114	B
123	CPS196	B+
857	CPS196	A+
857	CPS130	A+
456	CPS114	C
...

<i>SID</i>	<i>grade</i>
142	B-
142	B
123	B+
857	A+
857	A+
456	C
...	...

Questions about decomposition

20

- ❖ When to decompose
- ❖ How to come up with a correct decomposition (i.e., lossless join decomposition)

An answer: BCNF

21

- ❖ A relation R is in Boyce-Codd Normal Form if
 - For every non-trivial FD $X \rightarrow Y$ in R , X is a super key
 - That is, all FDs follow from “key \rightarrow other attributes”
- ❖ When to decompose
 - As long as some relation is not in BCNF
- ❖ How to come up with a correct decomposition
 - Always decompose on a BCNF violation
 - ☞ Then it is guaranteed to be a lossless join decomposition!

BCNF decomposition algorithm

22

- ❖ Find a BCNF violation
 - That is, a non-trivial FD $X \rightarrow Y$ in R where X is not a super key of R
- ❖ Decompose R into R_1 and R_2 , where
 - R_1 has attributes $X \cup Y$
 - R_2 has attributes $X \cup Z$, where Z contains all attributes of R that are in neither X nor Y
- ❖ Repeat until all relations are in BCNF

BCNF decomposition example

23

StudentGrade (*SID*, *name*, *email*, *CID*, *grade*)

BCNF violation: $SID \rightarrow name, email$

Another example

24

StudentGrade (*SID*, *name*, *email*, *CID*, *grade*)

BCNF violation:

Why is BCNF decomposition lossless 25

Given non-trivial $X \rightarrow Y$ in R where X is not a super key of R , need to prove:

- ❖ Anything we project always comes back in the join:
 $R \subseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R)$
 - Sure; and it doesn't depend on the FD
- ❖ Anything that comes back in the join must be in the original relation:
 $R \supseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R)$
 - Proof makes use of the fact that $X \rightarrow Y$

Recap 26

- ❖ Functional dependencies: a generalization of the key concept
- ❖ Non-key functional dependencies: a source of redundancy
- ❖ BCNF decomposition: a method for removing redundancies
 - BCNF decomposition is a lossless join decomposition
- ❖ BCNF: schema in this normal form has no redundancy due to FD's
