Relational Database Design Theory Part I

CPS 116 Introduction to Database Systems

Announcements (September 13)

- ❖ Homework #1 due this Thursday
- Course project assigned today
 - Choice of a "standard" or "open" course project
 - Two milestones (October 13 and November 10) and a final demo/report (December 6-13)

Motivation

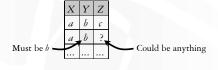
SID	name	CID
142	Bart	CPS116
142	Bart	CPS114
857	Lisa	CPS116
857	Lisa	CPS130

- How do we tell if a design is bad, e.g., StudentEnroll (SID, name, CID)?
 - This design has redundancy, because the name of a student is recorded multiple times, once for each course the student is taking
- How about a systematic approach to detecting and removing redundancy in designs?
 - Dependencies, decompositions, and normal forms

	2	

Functional dependencies

- * A functional dependency (FD) has the form $X \to Y$, where X and Y are sets of attributes in a relation R
- $* X \rightarrow Y$ means that whenever two tuples in R agree on all the attributes in X, they must also agree on all attributes in Y



FD examples

Address (street_address, city, state, zip)

- ❖ street_address, city, state → zip
- $\star zip \rightarrow city$, state
- \star zip, state \rightarrow zip?
- \star zip \rightarrow state, zip?

Keys redefined using FD's

A set of attributes K is a key for a relation R if

- $\star K \rightarrow \text{all (other) attributes of } R$
 - That is, *K* is a "super key"
- \diamond No proper subset of K satisfies the above condition
 - That is, K is minimal

Reasoning with FD's Given a relation R and a set of FD's \mathcal{F} ❖ Does another FD follow from \mathcal{F} ? lacksquare Are some of the FD's in ${\mathcal F}$ redundant (i.e., they follow from the others)? \star Is K a key of R? ■ What are all the keys of *R*? Attribute closure \star 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 $\{A_1, A_2, ...\}$ functionally determined by Z (that is, $Z \rightarrow A_1 A_2 ...$) * Algorithm for computing the closure • Start with closure = Z• If $X \to Y$ 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 StudentGrade (SID, name, email, CID, grade) (Not a good design, and we will see why later)

Example of computing closure $\star \mathcal{F}$ includes: ■ SID → name, email lacktriangledown email o SID SID, CID → grade **♦** { CID, email $}^+$ = ? \bullet email \rightarrow SID Add SID; closure is now { CID, email, SID } ❖ SID → name, email ■ Add name, email; closure is now { CID, email, SID, name } ❖ SID, CID \rightarrow grade • Add grade; closure is now all the attributes in Student Grade Using attribute closure Given a relation R and set of FD's \mathcal{F} ❖ Does another FD $X \to Y$ follow from \mathcal{F} ? lacksquare Compute X^+ with respect to ${\mathcal F}$ ■ If $Y \subseteq X^+$, then $X \to Y$ follow from \mathcal{F} \star 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 ❖ Armstrong's axioms ■ Reflexivity: If $Y \subseteq X$, then $X \to Y$ ■ Augmentation: If $X \to Y$, then $XZ \to YZ$ for any Z■ Transitivity: If $X \to Y$ and $Y \to Z$, then $X \to Z$ Rules derived from axioms • Splitting: If $X \to YZ$, then $X \to Y$ and $X \to Z$ • Combining: If $X \to Y$ and $X \to Z$, then $X \to YZ$

Using rules of FD's

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

- ❖ Does another FD $X \to Y$ follow from \mathcal{F} ?
 - Use the rules to come up with a proof
 - Example:
 - 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

❖ 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

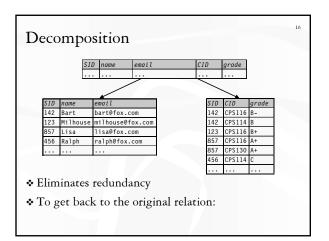
X	Y	Z
а	b	с 1
а	b	с2

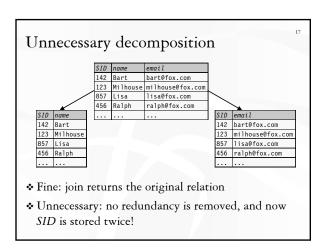
That α is always associated with b is recorded by multiple rows: redundancy, update anomaly, deletion anomaly

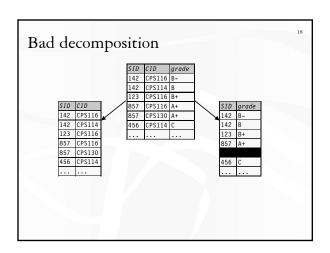
Example of redundancy

- * StudentGrade (SID, name, email, CID, grade)
- ❖ SID → name, email

SI	D	name	email	CID	grade
14	12	Bart	bart@fox.com	CPS116	B-
14	12	Bart	bart@fox.com	CPS114	В
12	23	Milhouse	milhouse@fox.com	CPS116	B+
85	57	Lisa	lisa@fox.com	CPS116	A+
85	57	Lisa	lisa@fox.com	CPS130	A+
45	6	Ralph	ralph@fox.com	CPS114	С



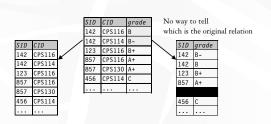




- \diamond Decompose relation R into relations S and T
 - $attrs(R) = attrs(S) \cup attrs(T)$
 - $\bullet \ S = \pi_{attrs(S)} (R)$
 - $\blacksquare \ T = \pi_{attrs(T)}(R)$
- ❖ The decomposition is a lossless join decomposition if, given known constraints such as FD's, we can guarantee that $R = S \bowtie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

Loss? But I got more rows!

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



Questions about decomposition

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

An answer: BCNF ❖ A relation *R* is in Boyce-Codd Normal Form if • For every non-trivial FD $X \to Y$ in R, X is a super key ■ That is, all FDs follow from "key → 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 (details next) Then it is guaranteed to be a lossless join decomposition! BCNF decomposition algorithm ❖ 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 StudentGrade (SID, name, email, CID, grade) BCNF violation: $SID \rightarrow name$, email

Student (SID, name, email)

BCNF

Grade (SID, CID, grade) BCNF

	1
Another example	
StudentGrade (SID, name, email, CID, grade)	
26.]
Why is BCNF decomposition lossless	
Given non-trivial $X \to Y$ 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 $X\to Y$	
11001 makes use of the fact that A / 1	-
	1
Recap	
 Functional dependencies: a generalization of the key concept 	
❖ Non-key functional dependencies: a source of	
redundancy	
❖ BCNF decomposition: a method for removing redundancies	
■ BNCF decomposition is a lossless join decomposition	

* BCNF: schema in this normal form has no

redundancy due to FD's