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

Functional dependencies

- A functional dependency (FD) has the form $X \rightarrow Y$, 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

X	Y	Z
a	*b*	*c*
a	*b*	?
…	…	…

Must be *b* (arrow to *b* in second row); Could be anything (arrow to ? in second row)

FD examples

Address (*street_address*, *city*, *state*, *zip*)

- *street_address*, *city*, *state* → *zip*
- *zip* → *city*, *state*
- *zip*, *state* → *zip*?
- *zip* → *state*, *zip*?

Keys redefined using FD's

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

- $K \rightarrow$ all (other) attributes of R
  - That is, K is a "super key"
- 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 $\mathcal{F}$?
  - 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

- 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 $\{A_1, A_2, \ldots\}$ functionally determined by Z (that is, $Z \rightarrow A_1 A_2 \ldots$)
- 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

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

(Not a good design, and we will see why later)

Example of computing closure

- $\mathcal{F}$ includes:
  - *SID* → *name*, *email*
  - *email* → *SID*
  - *SID*, *CID* → *grade*
- { *CID*, *email* }$^+$ = ?
- *email* → *SID*
  - Add *SID*; closure is now { *CID*, *email*, *SID* }
- *SID* → *name*, *email*
  - Add *name*, *email*; closure is now { *CID*, *email*, *SID*, *name* }
- *SID*, *CID* → *grade*
  - Add *grade*; closure is now all the attributes in *StudentGrade*

Using attribute closure

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

- 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

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

- 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	$c2$
…	…	…

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

Example of redundancy

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

SID	name	email	CID	grade
142	Bart	bart@fox.com	CPS116	B-
142	Bart	bart@fox.com	CPS114	B
123	Milhouse	milhouse@fox.com	CPS116	B+
857	Lisa	lisa@fox.com	CPS116	A+
857	Lisa	lisa@fox.com	CPS130	A+
456	Ralph	ralph@fox.com	CPS114	C
...	...	...	...	...

Decomposition

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	CPS116	B-
142	CPS114	B
123	CPS116	B+
857	CPS116	A+
857	CPS130	A+
456	CPS114	C
...	...	...

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

Unnecessary decomposition

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

SID	name
142	Bart
123	Milhouse
857	Lisa
456	Ralph
...	...

SID	email
142	bart@fox.com
123	milhouse@fox.com
857	lisa@fox.com
456	ralph@fox.com
...	...

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

Bad decomposition

SID	CID	grade
142	CPS116	B-
142	CPS114	B
123	CPS116	B+
857	CPS116	A+
857	CPS130	A+
456	CPS114	C
...	...	...

SID	CID
142	CPS116
142	CPS114
123	CPS116
857	CPS116
857	CPS130
456	CPS114
...	...

SID	grade
142	B-
142	B
123	B+
857	A+
[redacted]	
456	C
...	...

Lossless join decomposition

19

- 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 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!

20

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

SID	CID	grade
142	CPS116	B
142	CPS114	B-
123	CPS116	B+
857	CPS116	A+
857	CPS130	A+
456	CPS114	C
...	...	...

SID	CID
142	CPS116
142	CPS114
123	CPS116
857	CPS116
857	CPS130
456	CPS114
...	...

No way to tell
which is the original relation

SID	grade
142	B-
142	B
123	B+
857	A+
[redacted]	
456	C
...	...

Questions about decomposition

21

- 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 $X \rightarrow Y$ 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 (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

Another example

25

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

Why is BCNF decomposition lossless

26

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

27

- 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