Relational Database Design

CPS 216

Advanced Database Systems

Announcements

- Homework #1 out today
  - Due next Thursday in class
- Sign up to present a research paper
  - Sign-up sheet available in my office (D327) during my office hours
    - First-come, first-serve
  - Participation is voluntary
    - Allows you to drop your lowest homework grade
  - In groups of 2-4

2

Relational model: a review

- A database is a collection of relations (or tables)
- Each relation has a list of attributes (or columns)
- Each attribute has a domain (or type)
- Each relation contains a set of tuples (or rows)

3

Keys

- A set of attributes *K* is a key for a relation *R* if
  - In no instance of *R* will two different tuples agree on all attributes of *K*
    - That is, *K* is a "tuple identifier"
  - No proper subset of *K* satisfies the above condition
    - That is, *K* is minimal
- Example: *Student* (*SID*, *name*, *age*, *GPA*)
  - *SID* is a key of *Student*
  - {*SID*, *name*} is not a key (not minimal)

4

Schema versus data

Student

SID	*name*	*age*	*GPA*
142	Bart	10	2.3
123	Milhouse	10	3.1
857	Lisa	8	4.3
456	Ralph	8	2.3
…	…	…	…

- Is *name* a key of *Student*?
  - Yes? Seems reasonable for this instance
  - No! Student names are not unique in general
- Key declarations are part of the schema

5

More examples of keys

- *Enroll* (*SID*, *CID*)
  - {*SID*, *CID*}
- *Address* (*street_address*, *city*, *state*, *zip*)
  - {*street_address*, *city*, *state*}
  - {*street_address*, *zip*}

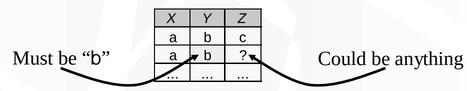
6

Usage of keys

- More constraints on data, fewer mistakes
- Look up a row by its key value
  - Many selection conditions are "key = value"
- "Pointers"
  - Example: *Enroll* (*SID*, *CID*)
    - *SID* is a key of *Student*
    - *CID* is a key of *Course*
    - *Enroll* "links" a *Student* row with a *Course* row
  - Many join conditions are "key = key value stored in another table"

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 tuple in R agree on all the attributes of X, they must also agree on all attributes of Y



FD examples

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

- *street_address*, *city*, *state* $\rightarrow$ *zip*
- *zip* $\rightarrow$ *city*, *state*
- *zip*, *state* $\rightarrow$ *zip*?
  - Trivial: LHS $\supseteq$ RHS
- *zip* $\rightarrow$ *state*, *zip*?
  - Non-trivial, but not completely: LHS $\cap$ RHS $\neq \varnothing$

➢ Completely non-trivial FD: LHS $\cap$ RHS = $\varnothing$

Keys redefined using FDs

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 FDs

Given a relation R and set of FDs $\boldsymbol{F}$

- Does another FD follow from $\boldsymbol{F}$?
  - Are some of the FDs in $\boldsymbol{F}$ redundant (because 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 FDs $\boldsymbol{F}$ that holds in R, and a set of attributes Z in R: The closure of Z with respect to $\boldsymbol{F}$ (denoted Z^+) is the set of all attributes functionally determined by Z
- Algorithm for computing the closure
  - Start with Z
  - If $X \rightarrow Y$ is in $\boldsymbol{F}$ and X is already in the closure, then also add Y to the closure
  - Repeat until you can't add anything more

A more complex example

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

- *SID* → *name*, *email*
- *email* → *SID*
- *SID*, *CID* → *grade*

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

Example of computing closure

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

Using attribute closure

Given a relation R and set of FDs $\mathcal{F}$

- 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?
  - Computer 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 FDs

- 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 FDs

Given a relation R and set of FDs $\mathcal{F}$

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

Non-key FDs

- 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
...	...	...

The fact that "a" is always associated with "b" is recorded in multiple rows: redundancy!

Problems with redundancy

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

SID → *name*, *email*

SID	*name*	*email*	*CID*	*grade*
142	Bart	bart@fox.com	CPS 216	B-
142	Bart	bart@fox.com	CPS 214	B
123	Milhouse	milhouse@fox.com	CPS 216	B+
857	Lisa	lisa@fox.com	CPS 216	A+
857	Lisa	lisa@fox.com	CPS 130	A+
456	Ralph	ralph@fox.com	CPS 214	C
…	…	…	…	…

- Wastes space
- Potential inconsistencies (update anomaly)

19

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	CPS 216	B-
142	CPS 214	B
123	CPS 216	B+
857	CPS 216	A+
857	CPS 130	A+
456	CPS 214	C
…	…	…

- Eliminates redundancy
- To get back to the original relation: $\bowtie$

20

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: now *SID* is stored twice!

21

Bad decomposition

SID	*CID*	*grade*
142	CPS 216	B-
142	CPS 214	B
123	CPS 216	B+
857	CPS 216	A+
857	CPS 130	A+
456	CPS 214	C
…	…	…

SID	*CID*
142	CPS 216
142	CPS 214
123	CPS 216
857	CPS 216
857	CPS 130
456	CPS 214
…	…

SID	*grade*
142	B-
142	B
123	B+
857	A+
~~857~~	~~A+~~
456	C
…	…

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

22

Lossless join decomposition

- Suppose that R is decomposed into S and T

  $\text{attrs}(R) = \text{attrs}(S) \cup \text{attrs}(T)$

  $S = \pi_{\text{attrs}(S)}(R)$

  $T = \pi_{\text{attrs}(T)}(R)$

- It is a lossless join decomposition if, given constraints such as FDs, we can guarantee

  $R = S \bowtie T$

23

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

SID	*CID*	*grade*
142	CPS 216	B
142	CPS 214	B-
123	CPS 216	B+
857	CPS 216	A+
857	CPS 130	A+
456	CPS 214	C
…	…	…

SID	*CID*
142	CPS 216
142	CPS 214
123	CPS 216
857	CPS 216
857	CPS 130
456	CPS 214
…	…

SID	*grade*
142	B-
142	B
123	B+
857	A+
~~857~~	~~A+~~
456	C
…	…

24

Questions about decomposition

- When to decompose
- How to come up with a correct decomposition

25

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 $\to$ 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's a lossless join decomposition!

26

BCNF decomposition algorithm

- Find a BNCF violation
  - That is, a non-trivial FD $X \to Y$ 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 $X \cup Z$ (*Z* contains all attributes of *R* that are in neither *X* nor *Y*)
- Repeat until all relations are in BNCF

27

BCNF decomposition example

StudentGrade (*SID*, *name*, *email*, *CID*, *grade*)
BCNF violation: *SID* $\to$ *name*, *email*

Student (*SID*, *name*, *email*)
BCNF

Grade (*SID*, *CID*, *grade*)
BCNF

28

Another example

StudentGrade (*SID*, *name*, *email*, *CID*, *grade*)
BCNF violation: *email* $\to$ *SID*

StudentID (*email*, *SID*)
BCNF

StudentGrade' (*email*, *name*, *CID*, *grade*)
BCNF violation: *email* $\to$ *name*

StudentName (*email*, *name*)
BCNF

Grade (*email*, *CID*, *grade*)
BCNF

29

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:
    $R \supseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R)$

30

Yet another example

- *Address* (*street_address*, *city*, *state*, *zip*)
  - *street_address*, *city*, *state* $\rightarrow$ *zip*
  - *zip* $\rightarrow$ *city*, *state*
- Keys
  - {*street_address*, *city*, *state*}
  - {*street_address*, *zip*}
- BCNF?
  - Violation: *zip* $\rightarrow$ *city*, *state*

31

To decompose, or not to decompose

$Address_1$ (*zip*, *city*, *state*)

$Address_2$ (*street_address*, *zip*)

- FDs in $Address_1$
  - *zip* $\rightarrow$ *city*, *state*
- FDs in $Address_2$
  - None!
- Hey, where is *street_address*, *city*, *state* $\rightarrow$ *zip*?
  - Cannot check it without joining $Address_1$ and $Address_2$ back together

32

"Elegant" solution

- Define the problem away!
- R is in Third Normal Form (3NF) if for every non-trivial FD $X \rightarrow A$, either
  - X is super key of R, or
  - A is a member of at least one key of R
- So *Address* is already in 3NF
- Tradeoff:
  - Can check all FDs in the decomposed relations
  - Might have some redundancy due to FDs

33

Recap

- Identifying tuples: keys
- Generalizing the key concept: FDs
- Non-key FDs: redundancy
- Avoiding redundancy: BCNF decomposition
- Preserving FDs: 3NF

34

What's next

- Another kind of dependency and normal form
- A comprehensive design example
- SQL basics

35