Relational Database Design Theory
Part I

CPS 116
Introduction to Database Systems

Announcements

- Homework #1 due this Thursday (Sept. 9) at midnight
- Course project assigned today
  - First milestone due September 30
- Details of (optional) student presentations will be available this Thursday
- Let me know if you still do not have a Gradiance or DB2 account

Motivation

- 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 $Y$

FD examples

Address (street_address, city, state, zip)
- street_address, city, state $\rightarrow$ zip
- zip $\rightarrow$ city, state
- zip, state $\rightarrow$ zip?
  - This is a trivial FD
  - Trivial FD: LHS $\supset$ RHS
- zip $\rightarrow$ state, zip?
  - This is non-trivial, but not completely non-trivial
  - Completely non-trivial FD: LHS $\cap$ RHS = $\emptyset$

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 functionally determined by $Z$
- 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

$StudentGrade (SID, name, email, CID, grade)$
- $SID \to name, email$
- $email \to SID$
- $SID, CID \to grade$
- Not a good design, and we will see why later

Example of computing closure

- $\mathcal{F}$ includes:
  - $SID \to name, email$
  - $email \to SID$
  - $SID, CID \to grade$
- $\{ CID, email \}^+ = ?$
  - $email \to SID$
    - Add $SID$; closure is now $\{ CID, email, SID \}$
  - $SID \to name, email$
    - Add $name, email$; closure is now $\{ CID, email, SID, name \}$
  - $SID, CID \to grade$
    - 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 $X \to Y$ follow from $\mathcal{F}$?
  - Compute $X^+$ with respect to $\mathcal{F}$
  - If $Y \subseteq X^+$, then $X \to Y$ 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 $X \to Y$
  - Augmentation: If $X \to Y$, then $XZ \toYZ$ 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 $X \rightarrow Y$ follow from $\mathcal{F}$?
  - Use the rules to come up with a proof
  - Example:
    - $\mathcal{F}$ includes:
      - $\text{SID} \rightarrow \text{name, email}$; $\text{email} \rightarrow \text{SID}$; $\text{SID, CID} \rightarrow \text{grade}$
      - $\text{email} \rightarrow \text{SID}$ (given in $\mathcal{F}$)
      - $\text{CID, email} \rightarrow \text{CID}$, $\text{SID}$ (augmentation)
      - $\text{SID, CID} \rightarrow \text{grade}$ (given in $\mathcal{F}$)
      - $\text{CID, email} \rightarrow \text{grade}$ (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 $X$

Example of redundancy
- StudentGrade ($\text{SID, name, email, CID, grade}$)
- $\text{SID} \rightarrow \text{name, email}$

Unnecessary decomposition
- Fine: join returns the original relation
- Unnecessary: no redundancy is removed, and now $\text{SID}$ is stored twice!

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

Bad decomposition
- Association between $\text{CID}$ and grade is lost
- Join returns more rows than the original relation
Lossless join decomposition

- Decompose relation $R$ into relations $S$ and $T$
  - $\text{attrs}(R) = \text{attrs}(S) \cup \text{attrs}(T)$
  - $S = \pi_{\text{attrs}(S)}(R)$
  - $T = \pi_{\text{attrs}(T)}(R)$
- The decomposition is a lossless join decomposition if, given 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 $R \subset S \bowtie T$

Questions about decomposition

- When to decompose
- How to come up with a correct decomposition (i.e., 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)
- Grade (SID, CID, grade)
  - BCNF

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
    - Then it is guaranteed to be a lossless join decomposition!

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

No way to tell which is the original relation
Another example

StudentGrade (SID, name, email, CID, grade)
BCNF violation: email → SID

StudentID (email, SID)
BCNF

StudentGrade' (email, name, CID, grade)
BCNF violation: email → name

StudentName (email, name)
BCNF

Grade (email, CID, grade)
BCNF

Why is BCNF decomposition lossless

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:
  \[ R \subseteq \pi_{XY}(R) \bowtie \pi_{XZ}(R) \]
  Yes, and it doesn’t depend on the FD

- Anything that comes back in the join must be in the original relation:
  \[ R \cong \pi_{XY}(R) \bowtie \pi_{XZ}(R) \]
  Proof makes use of the fact that \( X \rightarrow Y \)

Recap

- 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