Normalization

UVic C SC 370

Dr. Daniel M. German

Department of Computer Science

University of Victoria

July 30, 2003 Version: 1.1.0

Introduction

- What problems are caused by redundancy?
- What are functional dependencies?
- What are normal forms?
- What are the benefits of 3NF and BCNF?
- What is the process to decompose a relation into its normal forms?

Redundancy

- **Redundant storage**
- **Update anomalies**: all repeated data needs to be updated when one copy is updated
- **Insertion anomalies**: It might not be possible to store certain data unless some other, unrelated info is also stored
- **Deletion anomalies**: It may not be possible to delete certain info without losing some other, unrelated info

Example

ssn	name	lot	rating	hourWages	hoursWorked
123-45-789	Smiley	48	8	10	40
234-45-789	Smethurst	22	8	10	30
451-78-123	Guldu	35	5	7	30
457-89-023	Madayan	35	5	7	32

- The *hourly rate* is determined by the *rating* (this is an example of a functional dependency)
- Problems:
  - **Redundant Storage**: (8, 10) and (5,7) are repeated
  - **Update Anomalies**: The *hourWages* in the first tuple could be updated without making a similar update to the second tuple

Example...

ssn	name	lot	rating	hourWages	hoursWorked
123-45-789	Smiley	48	8	10	40
234-45-789	Smethurst	22	8	10	30
451-78-123	Guldu	35	5	7	30
457-89-023	Madayan	35	5	7	32

- Problems (cont):
  - **Insertion Anomalies**: We need to know the **hourWage** in order to insert a tuple (this could be fixed with a NULL value)
  - **Delete Anomalies**: If we delete all the tuples with a given (rating, hourWages) we might lose that association

Decomposition

- Functional dependencies can be used to **refine** the schema
- A relation is replaced with *smaller* relations
- **Decomposition of a relation schema R** consists in replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R.
- For example:

ssn	name	lot	rating	hoursWorked
123-45-789	Smiley	48	8	40
234-45-789	Smethurst	22	8	30
451-78-123	Guldu	35	5	30
457-89-023	Madayan	35	5	32

rating	hourWages
8	10
5	7

Problems related to Decomposition

- Do we need to decompose?
  - We use **normal forms** to answer this question
- What problems arise with a given decomposition? We are interested in the following properties of decompositions:
  - **Loss-less join**: Can we rebuild the original relation?
  - **Dependency preservation**: Do we preserve the ICs??

Functional Dependencies

- A **functional dependency** (FD) is a kind of IC that generalizes the concept of a key. Let R be a relation schema, with X and Y be nonempty sets of attributes in R. For an instance r of R, we say that the FD $X \rightarrow Y$ (X functionally determines Y) is satisfied if:

$$\forall t_1, t_2 \in r, t_1.X = t_2.X \implies t_1.Y = t_2.Y$$

Example of a Functional Dependency

- An instance that satisfies $AB \to C$:

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

- By looking at an instance, we can tell which FDs do not hold
- But we **cannot** tell which FDs will hold for any instance of the relation
- A primary key is a special case of a FD: If $X \to Y$ holds, where Y is the set of all attributes, then X is a superkey
- FDs should **hold** for any instance of a relation

Reasoning about FDs

- Given a set of FDs we can usually find additional FDs that also hold.
- Example: Given a key we can always find a superkey
- We say that an FD f **is implied** by a set F of FDs for relation schema R, if

$$\forall r \in R, \forall g \in F, g\ holds \implies f\ holds$$

Closure of a Set of FDs

- The set of all FDs implied by a given set F of FDs is called the **closure of F**, denoted by F^+
- Armstrong's Axioms for FDs. X, Y, Z are sets of attributes over a relation schema R.
  - **Reflexivity**: $Y \subseteq X \implies X \rightarrow Y$
  - **Augmentation**: $X \rightarrow Y \implies \forall Z, XZ \rightarrow YZ$
  - **Transitivity**: $X \rightarrow Y \wedge Y \rightarrow Z \implies X \rightarrow Z$

Closure...

- Armstrong's Axioms are sound and complete
- Two more rules non-essential rules:
  - **Union**: $X \to Y \wedge X \to Z \implies X \to YZ$
  - **Decomposition**: $X \to YZ \implies X \to Y \wedge X \to Z$

Examples

- What FDs can be inferred from: $A \rightarrow C$ and $B \rightarrow C$?
- We are given a relation $Contracts(contractid, supplierid, projectid, deptid, partid, qty, va$
  - $contractid$ is the key
  - A project purchases a given part using a single contract
  - A department purchases at most one part from a supplier

Attribute Closure

- The **attribute closure** X^+ with respect to a set F of FDs is the set of attributes A s.t. $X \rightarrow A$ can be inferred using Armstrong's Axioms.

- Algorithm to compute the closure:

  $closure = X;$

  repeat until there is no change: $\{$

  if $\exists U \rightarrow V \in F \; s.t. U \subseteq closure,$

  then set $closure = closure \bigcup V$

  $\}$

- This algorithm can be used to determine if $X \rightarrow Y$ is in F^+

Normal Forms

- We need to know, for a given schema, if it is a *good* design
- If we decompose, is the decomposition *good*?
- If a given relation schema is in a **normal form** we know some problems cannot arise.
- We are mainly interested in:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)

First Normal Form

- A relation is in **first normal form** if every field contains only **atomic** values (no lists nor sets)

Boyce-Codd Normal Form

- Let R be a schema
- F be a set of FDs that hold over R
- X be a subset of the attributes of R
- A be an attribute of R
- R is in **Boyce-Codd Normal Form**, if for every FD $X \to A$, one of the following is true:
  - $A \in X$ (it is a trivial FD) or
  - X is a superkey

Third Normal Form

- Let R be a schema
- F be a set of FDs that hold over R
- X be a subset of the attributes of R
- A be an attribute of R
- R is in **third normal form**, if for every FD $X \rightarrow A$, one of the following is true:
  - $A \in X$ (it is a trivial FD), or
  - X is a superkey, or
  - A is part of some key for R

Third Normal Form...

- In order to test if a relation is in 3NF, we need to find all the keys of the relation
- Finding all keys in a relation is NP complete!
- So is finding out if a relation is in 3NF
- Why do we want to use 3NF? Because **any** relation can be decomposed into a set of 3NF relations

Lossless join decomposition

- Let R be a schema
- F be a set of FDs that hold over R
- a decomposition of R into two schemas with attributes X and Y is **a lossless-join decomposition with respect to F** if $\forall r \in R$ that satisfies F:

$$\pi_x(r) \bowtie \pi_y(r) = r$$

Observation

- In general, for any relation $r \in R$ and set of attributes X, Y such that $X \cup Y$ = attributes(R)

$$r \subseteq \pi_x(r) \bowtie \pi_y(r)$$

Theorem 3

- Let R be a schema
- F be a set of FDs that hold over R
- a decomposition of R into two attributes sets R_1 and R_2 is lossless-join iff
  - $R_1 \cap R_2 \rightarrow R_1 \in F^+$, or
  - $R_1 \cap R_2 \rightarrow R_2 \in F^+$
- In other words, the attributes common to R_1 and R_2 should contain a key to either R_1 or R_2

Corollary 1

- If a $X \rightarrow Y$ holds over R, and $X \cap Y = \emptyset$, then the decomposition R into $R - Y$ and XY is loss-less join

Corollary 2

- If the relation R is decomposed into R_1 and R_2 is loss-less join, and then R_1 is decomposed into R_{11} and R_{12} loss-less join,
- then

$$R = (R_{11} \bowtie R_{12}) \bowtie R_2$$

Projection of F

- Let R be a relation schema,
- that is decomposed into 2 schemas with attribute sets X, Y
- Let F be a set of FDs over R
- The **projection** F on X is:

$$F_X = \{A \to B | A \to B \in F^+ \wedge A, B \subseteq X\}$$

Dependency Preserving Decomposition

- A decomposition of R with FDs F into schemas with atts X, Y is **dependency preserving** if

$$(F_X \cup F_Y)^+ = F^+$$

- In other words, we only need to enforce F_X and F_Y to guarantee that we are enforcing F^+

Normalization

- If a relation is **not in BCNF**, we can always find **a loss-less join decomposition into BCNF relations**
- If a relation is **not in 3NF**, we can find a **loss-less join, dependency preserving decomposition** of 3NF relations

Decomposition into BCNF

1. If R **is not in BCNF**, then using a FD $X \rightarrow A$ that is a violation of BCNF, decompose into $R - A$ and XA
2. If either$R - A$ or XA is not in BCNF, decompose them further by a recursive application of this algorithm

This decomposion is not FD preserving!

BCNF and Dependency Preservation

- Sometimes there is no BCNF decomposition that preserves FDs
- Example: SBD, with FD $SB \rightarrow D$, and $D \rightarrow B$
  - Decompose using $D \rightarrow B$
  - We can't preserve $SB \rightarrow D$
- One way to fix this is by adding a relation SBD with a FD $SB \rightarrow D$, but this defeats the purpose of decomposition

Minimal Cover of Set of FDs

- A minimal cover for a set of F of FDs is
  a set G of FDs such that:
  1. Every FD is of the form $X \rightarrow A$ for an attribute A
  2. $F^+ = G^+$
  3. If we obtain H such that $H \subsetneq G$ by deleting
     - a FD, or
     - an attribute from a FD, then

$$H^+ \neq G^+$$

- In other words, every FD is as small as possible, and every FD is required

Algorithm to obtain minimal cover

1. **Put the FDs in standard form** (one attribute in the right hand side)
2. **Minimize the left side of each FD**: For each FD in G, check each attribute in the left side to see if it can be deleted while preserving equivalence with F^+
3. **Delete redundant FDs**: Check each remaining FD in G to see if it can be deleted while preserving equivalence

There could be several minimal covers for a given set of FDs.

Dependency Preserving Decomp. into 3NF

- Given a relation R with a set of FDs **that is a minimal** cover,
- $R_1, R_2, ..., R_n$ is a loss-less join decomposition of R,
- F_i denotes the projection of F onto the attributes of R_i:
- Apply the following algorithm to find a Dependency Preserving Decomp. into 3NF:
  1. Identify the set N of dependencies in F that is not preserved
  2. For each FD $X \rightarrow A \in N$, create a relation schema XA and add it to the decomposition

Dep. Pres. Decomp. into 3NF...

- As an optimization, if N contains several FDs of with the same left side $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_n$, then replace it with a single $X \rightarrow A_1 \ldots A_n$ to produce a relation $XA_1 \ldots A_n$

Schema Refinement in Database Design

- Normalization can eliminate redundancy
- Conceptual design methodologies, such as ER arrive to an initial design
- But might not arrive to an optimal design
- Furthermore, they might not be able to express **all** FDs