Functional Dependencies & Normalization

Dr. Bassam Hammo

Redundancy and Normalisation

- Redundant Data
  - Can be determined from other data in the database
  - Leads to various problems
    - INSERT anomalies
    - UPDATE anomalies
    - DELETE anomalies

Redundancy and Normalisation

- Normalisation
  - Aims to reduce data redundancy
  - Redundancy is expressed in terms of dependencies
  - Normal forms are defined that do not have certain types of dependency

Functional Dependencies

- Redundancy is often caused by a functional dependency
- A functional dependency (FD) is a link between two sets of attributes in a relation
- We can normalise a relation by removing undesirable FDs

Functional Dependencies (FDs)

- A <u>functional dependency</u> X → Y holds over relation schema R if, for every allowable instance *r* of R:

$$t1 \in r,\ t2 \in r,\ \pi_X(t1) = \pi_X(t2)$$
$$\text{implies}\quad \pi_Y(t1) = \pi_Y(t2)$$

(where t1 and t2 are tuples; X and Y are sets of attributes)

- The attribute on the left side of the functional dependency is called the **determinant**, while the "→" reads as "**determines**"

Functional Dependencies

- In other words there exists **a functional dependency** between X and Y (**X → Y**), if whenever two rows of the relation have the same values for all the attributes in X, then they also have the same values for all the attributes in Y.
- Example:

  SID → DormName, Fee

  (CustomerNumber, ItemNumber, Quantity) → Price

- While a primary key is always a determinant, a determinant is not necessarily a primary key

Normalization

- Normalization eliminates **modification anomalies**
  - **Deletion anomaly**: deletion of a row loses information about two or more entities
  - **Insertion anomaly**: insertion of a fact in one entity cannot be done until a fact about another entity is added
- Anomalies can be removed by splitting the relation into two or more relations; each with a different, single theme
- However, breaking up a relation may create **referential integrity constraints**
- Normalization works through classes of relations called **normal forms**

FDs and Normalisation

- We define a set of 'normal forms'
  - Each normal form has fewer FDs than the last
  - Since FDs represent redundancy, each normal form has less redundancy than the last
- Not all FDs cause a problem
  - We identify various sorts of FD that do
  - Each normal form removes a type of FD that is a problem
  - We will also need a way to remove FDs

Reasoning About FDs

- Given some FDs, we can usually infer additional FDs:

  title → *studio, star* implies *title* → *studio* and *title* → *star*

  title → *studio* and *title* → *star* implies *title* → *studio, star*

  title → *studio*, *studio* → *star* implies *title* → *star*

  But,

  title, star → *studio* does NOT necessarily imply that

  title → *studio* or that *star* → *studio*

- An FD *f* is *implied by* a set of FDs *F* if *f* holds whenever all FDs in *F* hold.
- F^+ = *closure of F* is the set of all FDs that are implied by *F*. (includes "trivial dependencies")

Rules of Inference

- Armstrong's Axioms (X, Y, Z are <u>sets</u> of attributes):
  - *Reflexivity*: If $X \supseteq Y$, 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 $X \rightarrow Z$

- These are *sound* and *complete* inference rules for FDs!
  - i.e., using AA you can compute all the FDs in F+ and only these FDs.

- Some additional rules (that follow from AA):
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Rules of Inference

- Rules that follow from AA:
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

$X \rightarrow Y$,

$XX \rightarrow XY$ (Aug), so $X \rightarrow XY$

$X \rightarrow Z$,

$XY \rightarrow YZ$ (Aug) so $X \rightarrow YZ$ (Trans)

Rules of Inference

- Rules that follow from AA:
  - *Decomposition*: If $X \to YZ$, then $X \to Y$ and $X \to Z$

$X \to YZ$,

$YZ \to Y$ (Reflex), so $X \to Y$ (Trans)

Similar for $X \to Z$.

Attribute Closure

- Computing the closure of a set of FDs can be expensive. (Size of closure is exponential in # attrs!)
- Typically, we just want to check if a given FD $X \rightarrow Y$ is in the closure of a set of FDs F. An efficient check:
  - Compute *attribute closure* of X (denoted X^+) wrt F.
    X^+ = Set of all attributes A such that $X \rightarrow A$ is in F^+
    - $X^+ := X$
    - Repeat until no change: if there is an fd $U \rightarrow V$ in F such that U is in X^+, then add V to X^+
  - Check if Y is in X^+
  - Approach can also be used to find the keys of a relation.
    - If all attributes of R are in the closure of X then X is a superkey for R.
    - Q: How to check if X is a "candidate key"?

Attribute Closure (example)

$R = \{A, B, C, D, E\}$

$F = \{ B \rightarrow CD, D \rightarrow E, B \rightarrow A, E \rightarrow C, AD \rightarrow B \}$

- Is $B \rightarrow E$ in F^+ ?

  $B^+ = B$

  $B^+ = BCD$

  $B^+ = BCDA$

  $B^+ = BCDAE$ … Yes! and B is a key for R too!

- Is D a key for R?

  $D^+ = D$

  $D^+ = DE$

  $D^+ = DEC$ … Nope!

Attribute Closure (example)

R = {A, B, C, D, E}

F = { B →CD, D → E, B → A, E → C, AD →B }

- Is AD a key for R?

  AD^+ = AD

  AD^+ = ABD and B is a key, so Yes!

- Is AD a *candidate* key for R?

  A^+ = A, D+ = DEC

  A,D not keys, so Yes!

- Is ADE a *candidate* key for R?

  No! AD is a key, so ADE is a superkey, but not a candidate key

Normal Forms

- Any table of data is in 1NF if it meets the definition of a relation
- A relation is in 2NF if all its non-key attributes are dependent on all of the key (no **partial dependencies**)
  - If a relation has a single attribute key, it is automatically in 2NF
- A relation is in 3NF if it is in 2NF and has no **transitive dependencies**
- A relation is in BCNF if every determinant is a candidate key
- A relation is in fourth normal form if it is in BCNF and has no **multi-value dependencies**