Functional dependency theory

Introduction to Database Design 2011, Lecture 8
Overview

- Recalling normal forms
- Functional dependency theory
  - Computing closures of attribute sets
  - Canonical covers
  - Dependency preservation
Definition

• A **legal** instance of a database schema is an instance that does not break the rules of the real world

• **Definition.** A functional dependency $\alpha \rightarrow \beta$ holds if for all pairs of tuples $t, u$ in any legal instance:

\[
\text{if } t[\alpha] = u[\alpha] \text{ then } t[\beta] = u[\beta]
\]

• Here $\alpha, \beta$ denote sets of attributes

• Example: $\text{dept\_name} \rightarrow \text{budget}$ because:

\[
\text{if } t[\text{dept\_name}] = u[\text{dept\_name}] \text{ then } t[\text{budget}] = u[\text{budget}]
\]
Boyce-Codd normal form (BCNF)

• A table \( r(R) \) is in **BCNF** if for all functional dependencies \( \alpha \rightarrow \beta \) either
  - \( \beta \subseteq \alpha \) (\( \alpha \rightarrow \beta \) is trivial)
  - or \( \alpha \) is a superkey

• A schema is in **BCNF** if all tables are in BCNF

• A schema in BCNF does not allow for (most) redundancies

• Example of a non-BCNF schema:
  - instructor\((ID, \text{name}, \text{salary}, \text{dept\_name}, \text{building}, \text{budget})\)
Third normal form (3NF)

• A table r(R) is in **3NF** if for all functional dependencies α→β either
  - β ⊆ α (α→β is trivial)
  - or α is a superkey
  - or each A in β-α is part of a candidate key

• A schema is in **3NF** if all tables are in 3NF

• Any schema in BCNF is also in 3NF

Rasmus Ejlers Møgelberg
Motivation for 3NF

• Schemas in 3NF can contain some redundancies not found in schemas in BCNF

• Not always possible to reduce to BCNF in a dependency preserving way

• This means that 3NF schemas can be more efficient to work with

• See this weeks exercises for an illustration!
Functional-dependency theory
Functional dependency theory

- Formal theory of functional dependencies
- Rules for computing with these
- Can be used when showing that a database satisfies normal forms
- And for giving a formal definition of e.g. dependency preserving decomposition
- Also important for decomposition algorithms
Implied functional dependencies

- An implied functional dependency is one that follows from other stated ones

- Assume e.g.

\[
egin{align*}
ID & \rightarrow dept\_name \\
dept\_name & \rightarrow budget
\end{align*}
\]

- Then also

\[
ID \rightarrow budget
\]

- The definitions of the normal forms talk about all functional dependencies, including the implied ones
Armstrong’s axioms

- Axioms for deriving implied functional dependencies
  - **Reflexivity**: If $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$
  - **Augmentation**: If $\alpha \rightarrow \beta$ then $\alpha, \gamma \rightarrow \beta, \gamma$
  - **Transitivity**: If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then also $\alpha \rightarrow \gamma$

- Exercise: Check that these are valid

- **Theorem.** All functional dependencies can be derived from these

- Derived rule: If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta, \gamma$
Closures of attribute sets

- Suppose F is a set of functional dependencies
- Write $F^+$ for the set of functional dependencies implied by F
- To compute $F^+$ keep applying the axioms until stuck
- Suppose $\alpha$ is a set of attributes
- Write $\alpha^+$ for the largest set such that $\alpha \to \alpha^+$ is in $F^+$
Computing closures of attribute sets

- Set result = $\alpha$
- While result changes do
  - For all $\beta \rightarrow \gamma$ in $F$ do
    - if $\beta \subseteq$ result set result = result $\cup \gamma$
- $\alpha^+ = \text{result}$
Example

• If \( F \) is

\[
\begin{align*}
i_ID & \rightarrow dept_name \\
s_ID, dept_name & \rightarrow i_ID
\end{align*}
\]

• Then

\[
\begin{align*}
(s_ID, dept_name)^+ &= (s_ID, dept_name, i_ID) \\
(s_ID, i_ID)^+ &= (s_ID, dept_name, i_ID) \\
(i_ID)^+ &= (dept_name, i_ID)
\end{align*}
\]
• Consider schema \( R(A,B,C,D) \) with dependencies

\[
AB \rightarrow C \\
C \rightarrow D \\
D \rightarrow A
\]

• Compute all candidate keys

• Is \( R \) in BCNF?

• Is it in 3NF?

• (this is a typical exam exercise)
Implied functional dependencies and NFs

• The definitions of the normal forms talk about all functional dependencies, including the implied ones

• Consider $R(A,B), R'(A,C,D,E)$ with dependencies

\[ A \rightarrow B \]
\[ BC \rightarrow D \]

• $R'$ is not BCNF
Example

• Theorem.
  - Suppose F involves only attributes of r and suppose r satisfies the condition of BCNF for all functional dependencies in F.
  - Then it also satisfies the condition for all dependencies in $F^+$. 
• Suppose we want to test if an instance satisfies some functional dependencies

• Question: Can we find a minimal set of such dependencies that we need to check?

• Example:
  - Suppose we know that some instance satisfies $A \rightarrow B$ and $B \rightarrow C$
  - No need to test $A \rightarrow C$

• A canonical cover is such a minimal set

• Formal definition follows shortly
Extraneous attributes

- **Extraneous attributes** are attributes that can be removed from functional dependencies

- Examples
  - If $A \rightarrow C$ then $B$ is extraneous in $AB \rightarrow C$
  - If $A \rightarrow C$ then $C$ is extraneous in $AB \rightarrow CD$

- In the first case $B$ represents an unnecessary assumption
- In the second case $C$ represents something we already know
Extraneous attributes

- Consider functional dependency $\alpha \rightarrow \beta$ in $F$
  - $A$ in $\alpha$ is \textbf{extraneous} if $F$ implies
    $$(F - (\alpha \rightarrow \beta)) \cup ((\alpha - A) \rightarrow \beta)$$
  - $B$ in $\beta$ is \textbf{extraneous} if the following implies $F$
    $$(F - (\alpha \rightarrow \beta)) \cup (\alpha \rightarrow (\beta - B))$$

- Examples: $F = \{A \rightarrow C, AB \rightarrow C, AB \rightarrow CD\}$
  - $B$ is extraneous in $AB \rightarrow C$
  - $C$ is extraneous in $AB \rightarrow CD$
A canonical cover of F is a set of functional dependencies $F_c$ such that

- $F_c$ implies all dependencies in F
- F implies all dependencies in $F_c$
- No dependency of $F_c$ contains any extraneous attributes
- Each left hand side of $F_c$ is unique
• Compute the canonical cover of

\[ A \rightarrow BC \]
\[ B \rightarrow C \]
\[ A \rightarrow B \]
\[ AB \rightarrow C \]
Decomposing relations

• Suppose R is decomposed into R₁ ... Rₙ

• Decomposition is **lossless** if for all legal instances r of R

\[ r = \prod_{R_1}(r) \bowtie ... \bowtie \prod_{R_n}(r) \]

• Question:
  - To check functional dependencies, do we have to compute the join, or can we just test on each Rᵢ?

• If answer is yes, we say decomposition is **dependency preserving**
Dependency preservation formally

- Suppose \( R \) is decomposed into \( R_1 \ldots R_n \)
- Write \( F_i \) for set of dependencies in \( F^+ \) where all attributes in \( R_i \)
- Decomposition is dependency preserving if
  \[
  (F_1 \cup \ldots \cup F_n)^+ = F^+
  \]
- Example of a dependency preserving decomposition
  - instructor(\( ID, \) name, salary, dept_name)
  - department(dept_name, building, budget)
Test for functional dependencies

• Suppose R is decomposed into $R_1 \ldots R_n$ and decomposition is dependency preserving

• Compute for each i a canonical cover of $F_i$

• This is what we need to test for on each $R_i$

• Remember motivation:
  - Computing joins is expensive
  - Functional dependencies must be tested for every insertion and update
  - Insertions and updates need to be efficient
Learning objectives

• You should be able to
  - Decide if a schema is on BCNF
  - Decide if a schema is on 3NF
  - Decide if a decomposition is lossless
  - Decide if a decomposition is dependency preserving
  - Compute a canonical cover
Next time

- **BCNF decomposition**
  - Can always decompose a db design to BCNF in a lossless way

- **3NF decomposition**
  - Can always decompose into 3NF in a lossless and dependency preserving way