Design Theory: Functional Dependencies and Normal Forms, Part I

Instructor: Shel Finkelstein

Reference:
A First Course in Database Systems,
3rd edition, Chapter 3

Important Notices

- Gradiance #4 was assigned on Sunday, May 19, and is due on **Monday, May 27 by 11:59pm**.
- Lab4 assignment is due on **Sunday, June 2**, by 11:59pm.
  - Subject of Lab4 is Lecture 10 (Application Programming).
  - Lab4 will be discussed at Lab Sections.
  - Your solution should be submitted via Canvas as a zip file.
    - Canvas is used for both Lab submission and grading.
    - Late Lab Assignments will not be accepted.
    - Be sure that you post the correct file!
  - Load file for Lab4 has been/will be posted to Piazza.
    - You must use load file to do Lab4.
    - Load data helps with testing, but we won't post query solutions.
- See Small Group Tutoring website for LSS Tutoring with Chandler Hawkins.

Important Notices

CMPS 180 Final Exam is on **Monday June 10, 4:00 – 7:00pm**, in our usual classroom.

- **No** early/late Finals, **no** make-up Finals.
- **No** devices.
- Includes a Multiple Choice Section and a Longer Answers Section.
  - Bring Red Scantron sheets (ParSCORE form number f-1712) sold at Bookstore, and #2 pencils for Multiple Choice Section.
    - Ink and #3 pencils don't work.
- Covers entire quarter, with slightly greater emphasis on second half of quarter.
- You may bring in one double-sided 8.5 by 11 sheet, with anything that you can read unassisted printed or written on both sides of the paper.
  - **No sharing** of sheets is permitted.
  - Include name on top right of sheet. Sheets will be collected with Finals.
- You **must** show your UCSC ID at end of Final.
- Will post Practice Final from Spring 2017 (2 Sections) on Piazza.

Database Schema Design

- So far, we have learned database query languages:
  - SQL, Relational Algebra
- How can you tell whether a given database schema is “good” or “bad”?
- Design theory:
  - A set of design principles that allows one to decide what constitutes a “good” or “bad” database schema design.
  - A set of algorithms for modifying a “bad” design to a “better” one.

Example

- If we know that rank determines the salary scale, which is a better design? Why?

- Employees(eid, name, addr, rank, salary_scale)

OR

- Employees2(eid, name, addr, rank)
  Salary_Table(rank, salary_scale)

Lots of Duplicate Information

eid	name	addr	rank	salary_scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Lots of duplicate information
  - Employees who have the same rank have the same salary scale.

Update Anomaly

eid	name	addr	rank	salary_scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Update anomaly
  - If one copy of salary scale is changed, then all copies of that salary scale (of the same rank) have to be changed.

Insertion Anomaly

eid	name	addr	rank	salary_scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Insertion anomaly
  - How can we store a new rank and salary scale information if currently, no employee has that rank?
  - Use NULLS?

Deletion Anomaly

eid	name	addr	rank	salary_scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

- Deletion anomaly
  - If Hugh is deleted, how can we retain the rank and salary scale information?
  - Is using NULL a good choice?
    - (Why not?)

So What Would Be a Good Schema Design for this Example?

- salary_scale is dependent only on rank
  - Hence associating employee information such as name, addr with salary_scale causes redundancy.

- Based on the constraints given, we would like to refine the schema so that such redundancies **cannot** occur.

- Note however, that sometimes database designers **may choose** to live with redundancy in order to improve query performance.
  - Ultimately, a good design is depends on the query workload.
  - But understanding anomalies and how to deal with them is still important.

Functional Dependencies

- The information that rank determines salary_scale is a type of integrity constraint known as a *functional dependency (FD).*

- Functional dependencies can help us detect anomalies that may exist in a given schema.

- The FD “rank → salary_scale” suggests that

  Employees(<u>eid</u>, name, addr, rank, salary_scale)

  should be *decomposed* into two relations:

  Employees2(<u>eid</u>, name, addr, rank)

  Salary_Table(<u>rank</u>, salary_scale).

Meaning of an FD

- We have seen a kind of functional dependency before.
- Keys:
  - Emp(ssn, name, addr)
  - If two tuples agree on the ssn value, then they must also agree on the name and address values. (ssn → name, addr).

- Let **R** be a relation schema. A *functional dependency (FD)* is an integrity constraint of the form:

  X → Y (read as "X *determines* Y or X *functionally determines* Y")

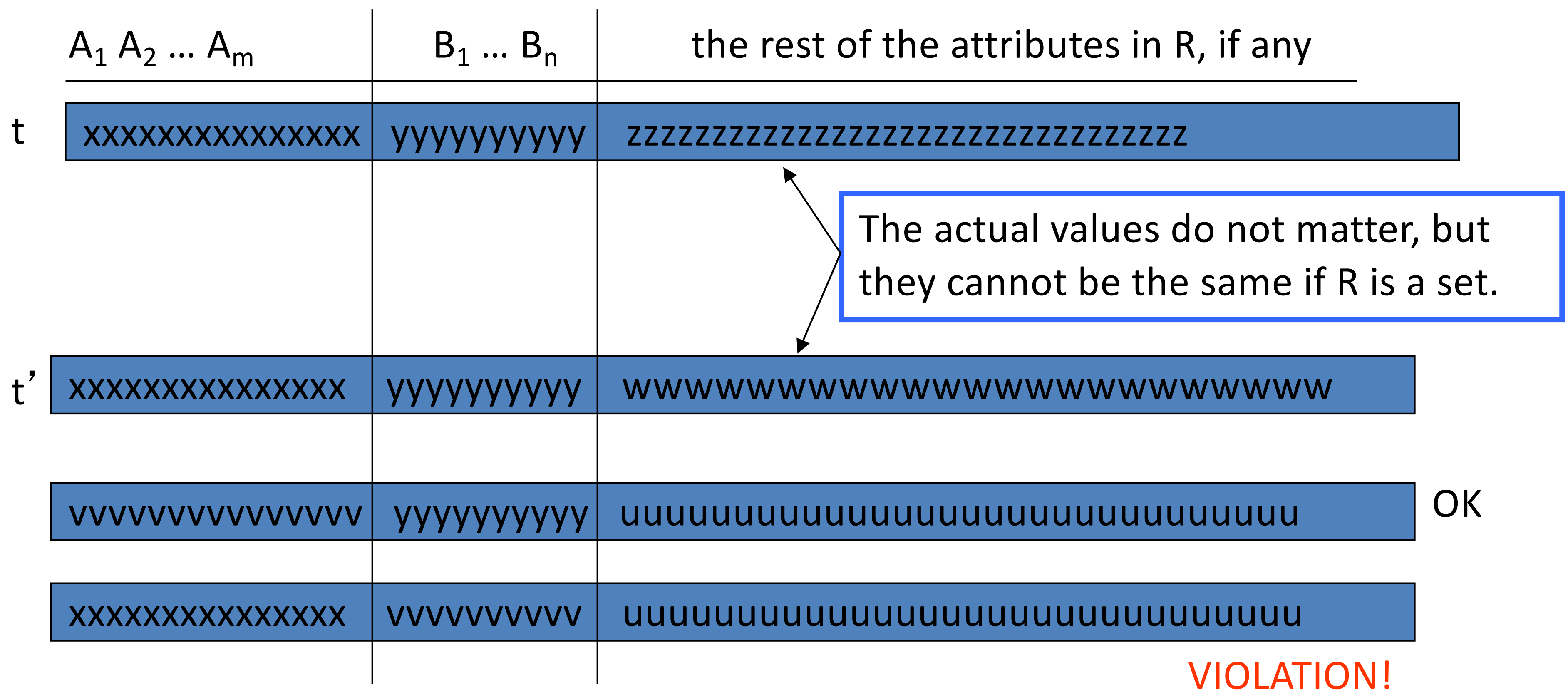
  where X and Y are non-empty subsets of attributes of **R**.
- A relation instance r of **R** *satisfies* the FD X → Y if

  for every pair of tuples t and t' in r, if t[X] = t'[X], then t[Y] = t'[Y]

Denotes the X value(s) of tuple t, i.e., project t on the attributes in X.

Illustration of the Semantics of an FD

- Relation schema R with the FD $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ where $\{ A_1, \ldots, A_m, B_1, \ldots, B_n \} \subseteq$ attributes(R).



More on Meaning of an FD

- Relation R satisfies X → Y
  - Pick any two (not necessarily distinct) tuples t and t’ of an instance r of R. If t and t’ agree on the X attributes, then they must also agree on the Y attributes.
  - The above must hold for *every possible instance* r of R.

- An FD is a statement about *all possible legal instances* of a schema. We <u>cannot</u> just look at an instance (or even at a set of instances) to determine which FDs hold.
  - Looking at an instance may enable us to determine that some FDs are <u>not satisfied</u>.

Reasoning about FDs

R(A,B,C,D,E)

Suppose A → C and C → E. Is it also true that A → E ?

In other words, suppose an instance r satisfies A → C and C → E, is it true that r must also satisfy A → E ?

YES

Proof: ?

Implication of FDs

- We say that a set $\mathcal{F}$ of FDs *implies* an FD F if for every instance r that satisfies $\mathcal{F}$, it must also be true that r satisfies F.
- Notation: $\mathcal{F} \vDash F$

- Note that just finding some instance(s) r such that r satisfies $\mathcal{F}$ and r also satisfies F is <u>not</u> sufficient to prove that $\mathcal{F} \vDash F$.

- How can we determine whether or not $\mathcal{F}$ implies F?

Armstrong's Axioms

- Use Armstrong's Axioms to determine whether or not $\mathcal{F} \vDash F$.

- Let X, Y, and Z denote sets of attributes over a relation schema R.

- **Reflexivity:** If Y ⊆ X, then X → Y.

  ssn, name → name

  - FDs in this category are called *trivial FDs.*
- **Augmentation**: If X → Y, then XZ → YZ for any set Z of attributes.

  ssn, name, addr → name addr
- **Transitivity**: If X → Y and Y → Z, then X → Z.

  If ssn → rank, and rank → sal_scale,

  then ssn → sal_scale.

Union and Decomposition Rules

- **Union**: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
- **Decomposition**: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$.

- Union and Decomposition rules are not essential. In other words, they can be derived using Armstrong's axioms.

- Derivation of the Union rule:

  (to fill in)

Union and Decomposition Rules

- **Union**: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
- **Decomposition**: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$.

- Union and Decomposition rules are not essential. In other words, they can be derived using Armstrong's axioms.

- Derivation of the Union rule:

  Since $X \rightarrow Z$, we get $XY \rightarrow YZ$ (augmentation)

  Since $X \rightarrow Y$, we get $X \rightarrow XY$ (augmentation)

  Therefore, $X \rightarrow YZ$ (transitivity)

Additional Rules

- Derivation of the Decomposition rule:
  (to fill in)

Additional Rules

- Derivation of the Decomposition rule:

  X → YZ (given)

  YZ → Y (reflexivity)

  YZ → Z (reflexivity)

  Therefore, X → Y and X → Z (transitivity).

- We use the notation $\mathcal{F} \vdash$ F to mean that *F can be derived from $\mathcal{F}$ using Armstrong's axioms*.
  - That's a lot of words, so we'll sometimes just read this as:
    "$\mathcal{F}$ generates F".
  - What was the meaning of $\mathcal{F} \vDash$ F ($\mathcal{F}$ implies F)?

Pseudo-Transitivity Rule

- **Pseudo-Transitivity**: If $X \rightarrow Y$ and $WY \rightarrow Z$, then $XW \rightarrow Z$.

- Can you derive this rule using Armstrong's axioms?

- Derivation of the Pseudo-Transitivity rule:

  (to fill in)

Pseudo-Transitivity Rule

- **Pseudo-Transitivity**: If $X \rightarrow Y$ and $WY \rightarrow Z$, then $XW \rightarrow Z$.

- Can you derive this rule using Armstrong's axioms?

- Derivation of the Pseudo-Transitivity rule:

  X -> Y and WY -> Z

  XW -> WY (augmentation)

  WY -> Z (given)

  Therefore XW -> Z (transitivity)

Completeness of Armstrong's Axioms

- **Completeness**: If a set $\mathcal{F}$ of FDs implies F, then F can be derived from $\mathcal{F}$ by applying Armstrong's axioms.
  - If $\mathcal{F} \models$ F, then $\mathcal{F} \vdash$ F.

    semantic syntactic

  - If $\mathcal{F}$ <u>implies</u> F, then one can prove F from $\mathcal{F}$ using Armstrong's axioms (i.e., $\mathcal{F}$ <u>generates</u> F).

For those familiar with Mathematical Logic:

- $\mathcal{F} \models$ F is "model-theoretic"
- $\mathcal{F} \vdash$ F is "proof-theoretic"

Soundness of Armstrong’s Axioms

- **Soundness**: If F can be derived from a set of FDs $\mathcal{F}$ through Armstrong' s axioms, then $\mathcal{F}$ implies F.
  - If $\mathcal{F} \vdash$ F, then $\mathcal{F} \vDash$ F.
    - That is, if $\mathcal{F}$ generates F, then $\mathcal{F}$ implies F.
  - Handwaving proof: If one can generate F from $\mathcal{F}$ using Armstrong’s axioms, then surely $\mathcal{F}$ implies F. (<u>Why?</u>)

- With Completeness and Soundness, we know that

  $\mathcal{F} \vdash$ F if and only if $\mathcal{F} \vDash$ F

  In other words, Armstrong' s axioms generate precisely *all* the FDs that must hold under $\mathcal{F}$ (all the axioms that $\mathcal{F}$ implies).

- Great! But how can we decide whether or not $\mathcal{F}$ implies F?

Closure of a Set of FDs $\mathcal{F}$

- Let $\mathcal{F}^+$ denote the set of all FDs implied by a given set $\mathcal{F}$ of FDs.
  - Also called the closure of $\mathcal{F}$.
- To decide whether $\mathcal{F}$ implies F, first compute $\mathcal{F}^+$, then see whether F is a member of $\mathcal{F}^+$.
- Example: Compute $\mathcal{F}^+$ for the set { $A \rightarrow B$, $B \rightarrow C$} of FDs.
- Trivial FDs
  - $A \rightarrow A$, $B \rightarrow B$, $C \rightarrow C$, $AB \rightarrow A$, $AB \rightarrow B$, $BC \rightarrow B$, $BC \rightarrow C$, $AC \rightarrow A$, $AC \rightarrow C$, $ABC \rightarrow A$, $ABC \rightarrow B$, $ABC \rightarrow C$, $ABC \rightarrow AB$, $ABC \rightarrow AC$, $ABC \rightarrow BC$, $ABC \rightarrow ABC$
- Transitivity (non-trivial FDs)
  - $AC \rightarrow B$ $AC \rightarrow A$ (trivial), $A \rightarrow B$ (given), so $AC \rightarrow B$ (transitivity).
  - $AB \rightarrow C$ $AB \rightarrow B$ (trivial), $B \rightarrow C$ (given), so $AB \rightarrow C$ (transitivity).
  - $A \rightarrow C$ $A \rightarrow B$ (given), $B \rightarrow C$ (given), so $A \rightarrow C$ (transitivity).

Expensive and tedious! Let's find a better way.

Attribute Closure Algorithm

- Let X be a set of attributes and $\mathcal{F}$ be a set of FDs. *The attribute closure X^+ with respect to $\mathcal{F}$* is the set of all attributes A such that $X \rightarrow A$ is derivable from $\mathcal{F}$.
  - That is, all the attributes A such that $\mathcal{F} \vdash X \rightarrow A$

  Input: A set X of attributes and a set $\mathcal{F}$ of FDs.

  Output: X^+

```
Closure = X;            // initialize Closure to equal the set X
repeat until no change in Closure {
  if there is an FD  U → V in F such that U ⊆ Closure,
  then Closure = Closure ∪ V;
}
return Closure;
```

If $A \in$ Closure (that is, if $A \in X^+$) , then $X \rightarrow A$.

More strongly, $\mathcal{F} \vdash X \rightarrow A$ if and only $A \in X^+$

FD Example 1 using Attribute Closure

- $\mathcal{F} = \{ A \to B, B \to C \}$.
- Question: Does $A \to C$?
- Compute A^+

- Closure = { A }
- Closure = { A, B } (due to $A \to B$)
- Closure = { A, B, C } (due to $B \to C$)
- Closure = { A, B, C }
  - no change, stop
- Therefore $A^+ = \{A, B, C\}$
- Since $C \in A^+$, answer YES.

FD Example 2 using Attribute Closure

- $\mathcal{F}$ = { AB $\rightarrow$ E, B $\rightarrow$ AC, BE $\rightarrow$ C }
- Question: Does BC $\rightarrow$ E?
- Compute BC^+

- Closure = { B, C }
- Closure = { A, B, C } (due to B $\rightarrow$ AC)
- Closure = { A, B, C, E } (due to AB $\rightarrow$ E)
- Closure = { A, B, C, E } (due to BE $\rightarrow$ C)
  - No change, so stop.
- Therefore BC^+ = {A,B,C,E}
- Since E $\in$ BC^+, answer YES.

A Better Algorithm for FDs

It's much easier to compute Attribute Closure X^+, rather than FD Closure $\mathcal{F}^+$

- To determine if an FD $X \rightarrow Y$ is implied by $\mathcal{F}$, compute X^+ and check if $Y \subseteq X^+$.
- Notice that computing Attribute Closure X^+ is less expensive (and less tedious) to compute than is FD Closure $\mathcal{F}^+$.

Correctness of Algorithm

- Is it correct?

  Prove that the algorithm indeed computes X^+.

  - Show that for any attribute $A \in X^+$, it is the case that $X \rightarrow A$ is derivable from $\mathcal{F}$.
  - Show if $X \rightarrow A$ is derivable from $\mathcal{F}$, then it must be that $A \in X^+$.

Proof of Correctness

Claim: If $A \in X^+$, then $\mathcal{F} \vdash X \rightarrow A$.

Proof: By induction on the number of iterations in the attribute closure algorithm.

(to fill in)

Soundness and Completeness of the Attribute Closure Algorithm

- Soundness: From previous slide, if $A \in X^+$, then $\mathcal{F} \vdash X \rightarrow A$.
  By the Soundness of Armstrong's axioms, it follows that $\mathcal{F} \vDash F$.

- Is it also true that if $\mathcal{F} \vDash F$, where F is the FD $X \rightarrow A$, then $A \in X^+$?

- Completeness.
  - Claim: If that if $\mathcal{F} \vDash F$, where F is the FD $X \rightarrow A$, then it must be the case that $A \in X^+$.
  - Proof by contradiction. Won't go through proof details.

Using Attribute Closure Algorithm to **Find All Superkeys/Keys** for Relation R, given Functional Dependencies $\mathcal{F}$

Attribute Closure algorithm can be modified to **find all superkeys and all candidate keys** for R, given Functional Dependencies $\mathcal{F}$.

- How?
  - Compute the closure of a single attribute in attr(R). Then compute the closure of every 2 attribute set, 3 attribute set, and so on.
  - If the closure of a set of attributes contains all attributes of relation R, then it is a *superkey* for R.
  - If no proper subset of those attributes has a closure that contains all attributes of the relation, then it is a *key*.

Using Attribute Closure Algorithm to Determine if a Set of Attributes X is a SuperKey/Key for Relation R, given Functional Dependencies $\mathcal{F}$

Attribute Closure algorithm can be modified to determine **if a set of attributes X is a superkey/key** for R, given Functional Dependencies $\mathcal{F}$.

- How?
  - Compute the attribute closure of X^+.
  - If X^+ = attr(R), then X is a *superkey*.
  - If no proper subset of X has a closure that contains all attributes of the relation, then X is a *key*.

Practice Homework 6

1. Let R(A,B,C,D,E) be a relation schema and let $\mathcal{F} = \{ AB \rightarrow E, B \rightarrow AC, BE \rightarrow C \}$ be a set of FDs that hold over R.

   a. Prove that $\mathcal{F} \models B \rightarrow E$ using Armstrong's axioms.
   b. Compute the closure of B. That is, compute B^+.
   c. Give a key for R. Justify why your answer is a key for R.
   d. Show an example relation that satisfies $\mathcal{F}$.
   e. Show an example relation that does not satisfy $\mathcal{F}$.

2. Let R(A,B,C,D,E) be a relation schema and let $\mathcal{F} = \{ A \rightarrow C, B \rightarrow AE, B \rightarrow D, BD \rightarrow C \}$ be a set of FDs that hold over R.

   a. Show that $B \rightarrow CD$ using Armstrong's axioms.
   b. Show a relation of R such that R satisfies $\mathcal{F}$ but R does not satisfy $A \rightarrow D$.
   c. Is AB a key for R?