CS143 Notes: Normalization Theory

INTRODUCTION

Main question

- How do we design "good" tables for a relational database?
  - Typically we start with ER and convert it into tables
  - Still, different people come up with different ER, and thus different tables. Which one is better? What design should we choose?

Book Chapters

(4th) Chapters 7.1-6, 7.8, 7.10
(5th) Chapters 7.1-6, 7.8

Warning

- The most difficult and theoretical part of the course. Pay attention!

MOTIVATION & INTUITION

⟨StudentClass(sid, name, addr, dept, cnum, title, unit) slide⟩

- **Q:** Is it a good table design?

- REDUNDANCY: The same information mentioned multiple times. Redundancy leads to potential anomaly.
  1. UPDATE ANOMALY: Only some information may be updated
     - **Q:** What if a student changes the address?
  2. INSERTION ANOMALY: Some information cannot be represented

- **Q:** What if a student does not take any class?

3. DELETION ANOMALY: Deletion of some information may delete others
   - **Q:** What if the only class that a student takes is cancelled?

- **Q:** Is there a better design? What tables would you use?

- **Q:** Any way to arrive at such table design more systematically?
  - **Q:** Where is the redundancy from?
    ⟨ Slide on "guessing" missing info ⟩
  - FUNCTIONAL DEPENDENCY: Some attributes are "determined" by other attrs
    * e.g., sid $\rightarrow$ (name, addr), (dept, cnum) $\rightarrow$ (title, unit)
    * When there is a functional dependency, we may have redundancy.
      · e.g., (301, James, 11 West) is stored redundantly. So is (CS, 143, database, 04).
  - DECOMPOSITION: When there is a FD, no need to store multiple instances of this relationship. Store it once in a separate table
    * ⟨Intuitive normalization of StudentClass table⟩
      StudentClass(sid, name, addr, dept, cnum, title, unit)
      FDs: sid$\rightarrow$(name, addr), (dept, cnum)$\rightarrow$(title, unit)
      1. sid $\rightarrow$ (name, addr): no need to store it multiple time. separate it out

      2. (dept, cnum) $\rightarrow$ (title, unit). separate it out

- Basic idea of table "normalization"

- Whenever there is a FD, the table may be "bad" (not in normal form)
- We use FDs to "split" or "decompose" table and remove redundancy
- We learn FUNCTIONAL DEPENDENCY and DECOMPOSITION to formalize this.

FUNCTIONAL DEPENDENCY

Overview

- The fundamental tool for normalization theory
- May seem dry and irrelevant, but bear with me. Extremely useful
- Things to learn
  - FD, trivial FD, logical implication, closure, FD and key, projected FD

Functional dependency $X \to Y$

- Notation: $u[X]$ - values for the attributes X of tuple u
  e.g, Assuming u = (sid: 100, name: James, addr: Wilshire), u[sid, name] = (100, James)
- FUNCTIONAL DEPENDENCY $X \to Y$
  - For any $u_1, u_2 \in R$, if $u_1[X] = u_2[X]$, then $u_1[Y] = u_2[Y]$
  - More informally, $X \to Y$ means that "no two tuples in R can have the same X values but different Y values"

    ⟨e.g., StudentClass(sid, name, addr, dept, cnum, title, unit)⟩
    * **Q:** sid → name?
    * **Q:** dept, cnum → title, unit?
    * **Q:** dept, cnum → sid?
  - Whether a FD is true or not depends on real-world semantics

    ⟨examples⟩

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_1	c_3

Q: AB $\rightarrow$ C. Is this okay?

Replace c_3 to c_1.

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_1	c_1

Q: AB $\rightarrow$ C. Is this okay?

NOTE: AB $\rightarrow$ C does not mean no duplicate C values.

Replace b_2 to b_1

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_3

Q: AB $\rightarrow$ C. Is this okay?

- TRIVIAL functional dependency: $X \rightarrow Y$ when $Y \subset X$
  - It is always true regardless of real world semantics
    (diagram)

- NON-TRIVIAL FD: $X \rightarrow Y$ when $Y \not\subset X$
  (diagram)

- COMPLETELY NON-TRIVIAL FD: $X \rightarrow Y$ with no overlap between X and Y
  (diagram)

We will focus on completely non-trivial functional dependency.

Implication and Closure

- LOGICAL IMPLICATION

  ex) $R(A, B, C, G, H, I)$
  F: $A \rightarrow B$, $A \rightarrow C$, $CG \rightarrow H$, $CG \rightarrow I$, $B \rightarrow H$ (set of functional dependencies)

- **Q:** Is $A \rightarrow H$ true under F?

  F LOGICALLY IMPLIES $A \rightarrow H$

  ⟨canonical database method to prove $A \rightarrow H$⟩

A	B	C	G	H	I
a_1	b_1	c_1	g_1	h_1	i_1
a_1				?	

  If ? $= h1$, then $A \rightarrow H$

  * **Q:** $AG \rightarrow I$?

- CLOSURE OF FD F: F^+

  F^+ : the set of all FD's that are logically implied by F.

- CLOSURE OF ATTRIBUTE SET X: X^+

  X^+: the set of all attrs that are functionally determined by X

  - **Q:** What attribute values do we know given (sid, dept, cnum)?

- CLOSURE X^+ COMPUTATION ALGORITHM

  ⟨X^+ computation algorithm slide⟩

  Start with $X^+ = X$

  Repeat until no change in X^+

  If there is $Y \rightarrow Z$ and $Y \subset X^+$, add Z to X^+

  ⟨example⟩

  $R(A, B, C, G, H, I)$ and $A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H$

  - **Q:** $\{A\}^+$?

- **Q:** $\{A, G\}^+$?

- FUNCTIONAL DEPENDENCY AND KEY
  - Key determines a tuple and functional dependency determines other attributes. Any formal relationship?
  - **Q:** In previous example, is (A, B) a key of R?
    $R(A, B, C, G, H, I)$ and $A \to B, A \to C, CG \to H, CG \to I, B \to H$
  - X is a KEY of R if and only if
    1. $X \to$ all attributes of R (i.e., $X^+ = R$)
    2. No subset of X satisfies 1 (i.e., X is minimal)
- PROJECTING FD
  $R(A, B, C, D) : A \to B, B \to A, A \to C$
  - **Q:** What FDs hold for $R'(B, C, D)$ which is a projection of R?
  - In order to find FD's after projection, we first need to compute F^+ and pick the FDs from F^+ with only the attributes in the projection.

DECOMPOSITION

- ⟨Remind the decomposition idea of StudentClass table⟩
- Splitting table $R(A_1, \ldots, A_n)$ into two tables, $R_1(A_1, \ldots, A_i)$ and $R_2(A_j, \ldots, A_n)$
  - $\{A_1, \ldots, A_n\} = \{A_1, \ldots, A_i\} \cup \{A_j, \ldots, A_n\}$
  - ⟨Conceptual diagram for $R(X, Y, Z) \to R_1(X, Y)$ and $R_2(Y, Z)$⟩

- **Q:** When we decompose, what should we watch out for?

LOSSLESS-JOIN DECOMPOSITION

- $R = R_1 \bowtie R_2$
- Intuitively, we should not lose any information by decomposing R
- Can reconstruct the original table from the decomposed tables
- **Q:** When is decomposition lossless?

⟨example⟩

cnum	sid	name
143	1	James
143	2	Elaine
325	3	Susan

  - **Q:** Decompose into S_1(cnum, sid), S_2(cnum, name). Lossless?

  - **Q:** Decompose into S_1(cnum, sid), S_2(sid, name). Lossless?

- DECOMPOSITION $R(X, Y, Z) \Rightarrow R_1(X, Y), R_2(X, Z)$ IS LOSSLESS IF $X \rightarrow Y$ OR $X \rightarrow Z$
  - That is, the shared attributes are the key of one of the decomposed tables
  - We can use FDs to check whether a decomposition is lossless

    Example: StudentClass(sid, name, addr, dept, cnum, title, unit)
    sid $\rightarrow$ (name,addr), (dept,cnum) $\rightarrow$ (title,unit)

    * **Q:** Decomposition into R_1(sid, name, addr), R_2(sid, dept, cnum, title, unit). Lossless?

BOYCE-CODD NORMAL FORM (BCNF)

FD, key & redundancy

- **Example:** StudentClass(sid, name, addr, dept, cnum, title, unit)
  - **Q:** sid $\rightarrow$ (name,addr). Does it cause redundancy?
  - After decomposition, Student(sid, name, addr)
    * **Q:** sid $\rightarrow$ (name,addr). Does it still cause redundancy?
    * **Q:** Why does the same FD cause redundancy in one case, but not in the other?
- In general, FD $X \rightarrow Y$ leads to redundancy if X DOES NOT CONTAIN A KEY.

BCNF definition

- R is in BCNF with regard to F, iff for every non-trivial $X \rightarrow Y$, X contains a key
- "Good" table design (no redudancy due to FD)
- **Q:** Class(dept, cnum, title, unit). dept,cnum$\rightarrow$title,unit.
  - **Q:** Intuitively, is it a good table design? Any redundancy? Any better design?
  - **Q:** Is it in BCNF?
- **Q:** Employee(name, dept, manager). name$\rightarrow$dept, dept$\rightarrow$manager.
  - **Q:** What is the English interpretation of the two dependencies?
  - **Q:** Intuitively, is it a good table design? Any redundancy? Better design?

– **Q:** Is it in BCNF?

- **Remarks:** Most times, BCNF tells us when a design is "bad" (due to redundancy from functional dependency.

BCNF normalization algorithm

- Decomposing tables until all tables are in BCNF
  - For each FD $X \to Y$ that violates the condition, separate those attributes into another table to remove redundancy.
  - We also have to make sure that this decomposition is lossless.
- **Algorithm**

  For any R in the schema
  If non-trivial $X \to Y$ holds on R, and if X does not have a key

  1. Compute X^+ (X^+: closure of X)
  2. Decompose R into $R_1(X^+)$ and $R_2(X, Z)$ // X is common attributes where Z is all attributes in R except X^+

  Repeat until no more decomposition
- **Example:** ClassInstructor(dept, cnum, title, unit, instructor, office, fax)
  instructor → office, office → fax
  (dept, cnum) → (title, unit), (dept, cnum) → instructor.
  - **Q:** What is the English interpretation of the two dependencies?
  - **Q:** Intuitively, is it a good table design? Any redundancy? Better design?
  - **Q:** Is it in BCNF?
  - **Q:** Normalize it into BCNF using the algorithm.

NOTE: The algorithm guarantees lossless join decomposition, because after the decomposition based on $X \to Y$, X becomes the key of one of the decomposed table

- **Example:** $R(A, B, C, G, H, I)$, $A \to B, A \to C, G \to I, B \to H$. Convert to BCNF.

- **Q:** Does the algorithm lead to a unique set of relations?

  ⟨e.g., $R(A, B, C), A \to C, B \to C$⟩

  Q: What if we start with $A \to C$?

  Q: What if we start with $B \to C$?

- **Q:** $R_1(A, B), R_2(B, C, D)$ with $A \to B, B \to A, A \to C$. Are R_1 and R_2 in BCNF?

  NOTE: We have to check all implied FD's for BCNF, not just the given ones.

MULTI-VALUED DEPENDENCY AND 4NF

Motivation

- **Example:** Classes, students and TAs. Every TA is for every student.
  cnum: 143, TA: (tony, james), sid: (100, 101, 103).
  cnum: 248, TA: (tony, susan), sid: (100, 102).

  ⟨entity-relationship diagram⟩

  ⟨Potentially good table design⟩

cnum	ta
143	tony
143	james
248	tony

cnum	sid
143	100
143	101
143	103
248	100
248	102

⟨Potentially bad table design⟩

cnum	ta	sid
143	tony	100
143	tony	101
143	tony	103
143	james	100
143	james	101
143	james	103
248	tony	100
248	tony	102

- **Q:** Does it have redundancy?

- **Q:** Does it have a FD?

- **Q:** Is it in BCNF?

- **Q:** Where does the redundancy come from?

  Note:

  * Two indepedent information (cnum, ta) and (cnum, sid) are put together in one table.
  * No direct connection between ta and student. The connection is through class.

- **Q:** How can we detect this kind of "bad" design?

  * **Q:** Assume that we have seen only the first 7 tuples in the table. Just based on these, can we "predict" that the table should also contain the 8th tuple?

* Note:
  * In each class, every ta appears with every student (ta $\times$ student)
  * For C_1, if TA_1 appears with S_1 and TA_2 appears with S_2, then TA_1 also appears with S_2.

Multivalued dependency $X \twoheadrightarrow Y$

- Definition: for every tuple u, $v \in R$:
  - If $u[X] = v[X]$, then there exists a tuple w such that:
    1. $w[X] = u[X] = v[X]$
    2. $w[Y] = u[Y]$
    3. $w[Z] = v[Z]$ where Z is all attributes in R except X and Y

  ⟨Explanation using canonical database⟩

  - MVD requires that tuples of a certain form exist: "tuple generating dependency"

  ⟨Explation using Y circle diagram⟩

  - $X \twoheadrightarrow Y$ means that if two tuples in R agree on X, we can swap Y values of the tuples and the two new tuples should still exist in R.
- **Example:** Class(cnum, ta, sid). cnum $\twoheadrightarrow$ ta? cnum $\twoheadrightarrow$ sid?
- COMPLEMENTATION RULE
  - $X \twoheadrightarrow Y$, then $X \twoheadrightarrow Z$ where Z is all attributes in R except X and Y
  - Proof: swapping Y is the same as swapping Z
- TRIVIAL MVD
  - $X \twoheadrightarrow Y$ is trivial MVD if
    1. $Y \subset X$ or
    2. $X \cup Y = R$

    ⟨Prove by canonical database⟩

	X	Y	Z
	x_1	y_1	z_1
	x_1	y_2	z_2
$\rightarrow$	x_1	y_1	z_2

FOURTH NORMAL FORM (4NF)

- Definition: R is in 4NF if for every nontrivial FD $X \to Y$ or MVD $X \twoheadrightarrow Y$, X contains a key
- **Q:** Relationship beteen BCNF and 4NF?

  $R(A_1, A_2, \ldots)$
  $X_1 \to Y_1$
  $X_2 \to Y_2$
  $\ldots$
  $Z_1 \twoheadrightarrow U_1$
  $Z_2 \twoheadrightarrow U_2$
  $\ldots$

  In BCNF, all X_i should contain a key
  In 4NF, all X_i's and Z_i's should contain a key
  ⟨Vann diagram of BCNF and 4NF⟩

- 4NF removes redundancy from MVD
  - 4NF: no redundancy from MVD and FD.
  - BCNF: no redundancy from FD.

4NF DECOMPOSITION

- First, using all functional dependencies, normalize tables into BCNF. Once the tables are in BCNF, apply the following algorithm to normalize them further into 4NF.
- **Algorithm**

  For any R in the schema
    If non-trivial $X \twoheadrightarrow Y$ holds on R, and if X does not contain a key
      Decompose R into $R_1(X, Y)$ and $R_2(X, Z)$ // X is common attributes
      where Z is all attributes in R except (X, Y)
  Repeat until no more decomposition

- **Example**: Class(cnum, ta, sid). cnum $\twoheadrightarrow$ ta.
  - **Q:** It is a good table design? Any better design?
  - **Q:** It is in 4NF?

- **Q:** Normalize into 4NF

- **Example**: Class(dept, cnum, title, ta, sid, sname). dept,cnum $\rightarrow$ title
  sid $\rightarrow$ sname
  dept,cnum,title $\twoheadrightarrow$ ta

  - **Q:** It is a good table design? Any better design?

  - **Q:** It is in 4NF?

  - **Q:** Normalize into 4NF

      * Note: dept,cnum,title $\twoheadrightarrow$ ta does not hold once the table is decomposed, but dept,cnum $\twoheadrightarrow$ ta should still hold.
      * In general, we have to compute the implied MVDs after decomposition.
        · The derivation can be done using 8 inference rules for MVD
        ⟨inference rule slides⟩
      * Formal derivation of implied MVDs is not a major topic of the class. For homeworks and exams, just derive them identify what are "intuitively" implied.

SIMPLIFIED 4NF DEFINITION

- MVD AS A GENERALIZATION OF FD
  - If $X \rightarrow Y$, then $X \twoheadrightarrow Y$
  - Proof: If X $\rightarrow$ Y, swapping Y values does not create new tuples.

    ⟨Prove by canonical database⟩

	X	Y	Z
	x_1	y_1	z_1
	x_1	y_2	z_2
$\rightarrow$	x_1	y_1	z_2

- Simplified definition of 4NF: R is in 4NF if for every nontrivial MVD $X \twoheadrightarrow Y$, X contains a key
  - Since $X \to Y$ implies $X \twoheadrightarrow Y$, this is sufficient.

GOOD TABLE DESIGN IN PRACTICE

- Normalization splits tables to reduce redundancy (based on FDs, MVDs).
- However, splitting tables has negative performance implication

  Example: Instructor: name, office, phone, fax
  name → office, office → (phone,fax)

  (design 1) Instructor(name, office, phone, fax)
  (design 2) Instructor(name, office), Office(offce, phone, fax)

  Q: Retrieve (name, office, phone) from Instructor. Which design is better?

- As a rule of thumb, start with normalized tables and merge them if performance is not good enough

DEPENDENCY-PRESERVATION AND 3NF

- FD is a type of constraint.
- We sometimes may want to make sure that an update does not violate FDs.
  ⟨eg, R(office, fax), office → fax. How do we enforce the FD?⟩

  ⟨eg, $R_1(A, B), R_2(A, C)$. $B \to C$. How to enforce it?⟩

  – Note: FD violation checking can be very expensive. Can we make sure we can enforce FD's without joins?

  ⟨eg, $R_1(A, B), R_2(B, C)$. $A \to B, B \to C, A \to C$. How to enforce them?⟩

  – No need to check $A \to C$ directly ($A \to B, B \to C$ implies $A \to C$).
  "Local FD checking" possible

DEPENDENCY-PRESERVING DECOMPOSITION

- When we can enforce FD's LOCALLY on each tables without joins after decomposition.
- **Q:** Does BCNF normalization lead to dependency-preserving decomposition?
- **Example 1:** ClassInstructor(dept, cnum, title, unit, instructor, office, fax)
  instructor → office
  office → fax
  (dept, cnum) → title, unit
  (dept, cnum) → instructor

  ⇒ BCNF: R_1(dept, cnum, title, unit, instructor), R_2(instructor, office), R_3(office, fax)

  – **Q:** Is it dependency preserving? Is it generally true?

- **Example 2:** R(street, city, zip), (street,city) $\rightarrow$ zip, zip $\rightarrow$ city
  $\Rightarrow$ BCNF: R_1(street, zip), R_2(zip, city).
  - **Q:** Is it dependency preserving?
- **Q:** Is there an alternative definition of normal form that guarantees dependency preservation?

THIRD-NORMAL FORM (3NF)

- R is in 3NF with regard to F, iff for every non-trivial X $\rightarrow$ Y either
  1. X contains a key OR
  2. Each attr in Y is a member of some key
- Example: R(street, city, zip). (street,city) $\rightarrow$ zip, zip $\rightarrow$ city
  - **Q:** BCNF?
  - **Q:** 3NF?
- IMPORTANT THEOREM: We can always decompose a relation into 3NF relations such that we can check all FD's locally.
  - This theorem essentially mean that 3NF is a normal form allowing dependency preservation.
  - The second condition of 3NF, "Y is a memeber of some key", is the "minimal relaxation" of BCNF to allow dependency preservation.
    * Do not ask me why. I do not have the intuition for the second condition.
- There exists "3NF Synthesis algorithm" that decomposes a table into a set of 3NF tables that preserve dependency.

COMPARISON OF NORMAL FORMS

- Comparison of 3NF and BCNF
  - **Q:** If R is in BCNF, is it in 3NF?
  - **Q:** If R is in 3NF, is it in BCNF?
- Relationship of 3NF, 4NF, BCNF
  - 4NF $\rightarrow$ BCNF $\rightarrow$ 3NF
    $\langle$vann diagram for three normal forms$\rangle$
- BCNF
  - Removes all redundancy due to FD
  - Does not guarantee dependency-preservation
- 4NF
  - Removes all redundancy due to FD and MVD
  - Does not guarantee dependency-preservation
- 3NF
  - Does not remove all redundancy due to FD or MVD
  - Dependency-preserving decomposition is always possible
  - 3NF is not often used in practice

3NF SYNTHESIS ALGORITHM

- We first need to learn a concept called CANNONICAL COVER.
  Example) F: $A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C$.
  - **Q:** Is $A \rightarrow BC$ necessary?
  - **Q:** Is $AB \rightarrow C$ necessary?
- CANONICAL COVER of F: F_C.
  - A minimum set of FD that is equivalent to F
    * If we delete any FD in F_C, no more equivalence.
    * If we delete any attribute from a FD in F_C, no more equivalence.
    * No two FDs in F_C have the same left-hand side

  ⟨eg, R(A,B,C). F: A → BC, B → C. Is F a canonical cover?⟩

  ⟨eg, R(A,B,C) F: A → B, B → AC. Is F a canonical cover?⟩

  Q: What is the canonical cover for StudentClass?
- CANONICAL COVER COMPUTATION ALGORITHM
  1. Split FDs into non-trivial FDs that have only one attribute on the right side
     - eg) ABC → ADE ⇒ ABC → D, ABC → E
  2. Minimize the left side of each FD
     - For each FD, check each attribute in the left side to see if it can be deleted while preserving equivalence
     - eg) AB → D, B → D ⇒ B → D, B → D
  3. Delete redundant FDs
     - eg) A → B, B → C, A → C ⇒ A → B, B → C
  4. Merge FDs with the same left-hand side
     - AB → C, AB → D ⇒ AB → CD

- **Example:**
  R(A, B, C, G, H, I).
  F: A $\rightarrow$ B, A $\rightarrow$ C, CG $\rightarrow$ H, CG $\rightarrow$ I, B $\rightarrow$ H.
  What is a canonical cover of F?

- Note
  - Canonical cover may not be unique
  - Most of the time, people easily find a canonical cover by intuition

3NF SYNTHESIS ALGORITHM

- **Algorithm**
  1. For every $X \rightarrow Y$ in F_C
     Create a relation $R_i(X, Y)$
  2. If None of R_i's contains a key of the original relation
     Create $R_i(X)$ where X is the key of the original relation
  3. If R_i and R_j have a common key
     Merge R_i and R_j
  4. Remove redundant relation

- ⟨eg, Student(sid, name, street, city, zip)⟩
  - Fc: sid $\rightarrow$ (name,street,city), (street, city) $\rightarrow$ zip, zip $\rightarrow$ city
  - Decompose into 3NF
    1. R_1(sid, name, street, city), R_2(zip, street, city), R_3(zip, city)
    2. -
    3. -
    4. Remove R_3

- ⟨eg, $R(A, B, C, D, E, F)$⟩
  - Fc: $A \rightarrow B, B \rightarrow D, C \rightarrow DE, D \rightarrow C$
  - Decompoes into 3NF
    1. $R_1(A, B), R_2(B, D), R_3(C, D, E), R_4(D, C)$
    2. Add $R_5(A, F)$
    3. Merge R_3 and R_4
    4. -