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?

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 $X \rightarrow Y$

• Notation: $u[X]$ - values for the attributes $X$ of tuple $u$
e.g., Assuming $u = (\text{sid: 100, name: James, addr: Wilshire})$, $u[\text{sid, name}] = (100, \text{James})$
• FUNCTIONAL DEPENDENCY $X \rightarrow Y$
  – For any $u_1, u_2 \in R$, if $u_1[X] = u_2[X]$, then $u_1[Y] = u_2[Y]$
  – More informally, $X \rightarrow Y$ 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
---|---|---
a₁ | b₁ | c₁
a₁ | b₂ | c₂
a₂ | b₁ | c₃

Q: AB \rightarrow C. Is this okay?

Replace c₃ to c₁.
A | B | C
---|---|---
a₁ | b₁ | c₁
a₁ | b₂ | c₂
a₂ | b₁ | c₁

Q: AB \rightarrow C. Is this okay?

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

Replace b₂ to b₁
A | B | C
---|---|---
a₁ | b₁ | c₁
a₁ | b₁ | c₂
a₂ | b₁ | c₃

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 $A \rightarrow H$)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>$g_1$</td>
<td>$h_1$</td>
<td>$i_1$</td>
<td>?</td>
</tr>
</tbody>
</table>

If $? = h_1$, 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?

\[
\begin{array}{ccc}
\text{cnum} & \text{sid} & \text{name} \\
143 & 1 & \text{James} \\
143 & 2 & \text{Elaine} \\
325 & 3 & \text{Susan} \\
\end{array}
\]

- Q: Decompose into \( S_1(\text{cnum}, \text{sid}) \), \( S_2(\text{cnum}, \text{name}) \). Lossless?

- Q: Decompose into \( S_1(\text{cnum}, \text{sid}) \), \( S_2(\text{sid}, \text{name}) \). 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(\( \text{sid, name, addr, dept, cnum, title, unit} \))

\[
\text{sid} \rightarrow (\text{name,addr}), (\text{dept,cnum}) \rightarrow (\text{title,unit})
\]

* Q: Decomposition into \( R_1(\text{sid, name, addr}) \), \( R_2(\text{sid, dept, cnum, title, unit}) \). Lossless?
BOYCE-CODD NORMAL FORM (BCNF)

FD, key & redundancy

- **Example:** StudentClass(sid, name, addr, dept, cnum, title, unit)
  - Q: sid → (name, addr). Does it cause redundancy?

  - After decomposition, Student(sid, name, addr)
    * Q: sid → (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 redundancy due to FD)
- Q: Class(dept, cnum, title, unit). dept, cnum → 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 → dept, dept → 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 $X \rightarrow Y$ 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 $X \rightarrow Y$ 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 $\rightarrow$ office, office $\rightarrow$ fax
  (dept, cnum) $\rightarrow$ (title, unit), (dept, cnum) $\rightarrow$ 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 decomposi-
tion based on \( X \rightarrow Y \), \( X \) becomes the key of one of the decomposed table

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

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

  \( \langle \text{e.g., } R(A, B, C), A \rightarrow C, B \rightarrow C \rangle \)

  **Q:** What if we start with \( A \rightarrow C \)?

  **Q:** What if we start with \( B \rightarrow C \)?

- **Q:** \( R_1(A, B), R_2(B, C, D) \) with \( A \rightarrow B \), \( B \rightarrow A \), \( A \rightarrow C \). 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), sid: (100, 102).

  (entity-relationship diagram)

(Potentially good table design)
Potentially bad table design

<table>
<thead>
<tr>
<th>cnum</th>
<th>ta</th>
<th>sid</th>
</tr>
</thead>
<tbody>
<tr>
<td>143</td>
<td>tony</td>
<td>143</td>
</tr>
<tr>
<td>143</td>
<td>james</td>
<td>143</td>
</tr>
<tr>
<td>248</td>
<td>tony</td>
<td>248</td>
</tr>
<tr>
<td></td>
<td></td>
<td>248</td>
</tr>
</tbody>
</table>

(Q: Does it have redundancy?)

<table>
<thead>
<tr>
<th>cnum</th>
<th>ta</th>
<th>sid</th>
</tr>
</thead>
<tbody>
<tr>
<td>248</td>
<td>tony</td>
<td>100</td>
</tr>
<tr>
<td>248</td>
<td>tony</td>
<td>102</td>
</tr>
<tr>
<td>143</td>
<td>tony</td>
<td>100</td>
</tr>
<tr>
<td>143</td>
<td>tony</td>
<td>101</td>
</tr>
<tr>
<td>143</td>
<td>tony</td>
<td>103</td>
</tr>
<tr>
<td>143</td>
<td>james</td>
<td>100</td>
</tr>
<tr>
<td>143</td>
<td>james</td>
<td>101</td>
</tr>
<tr>
<td>143</td>
<td>james</td>
<td>103</td>
</tr>
</tbody>
</table>

(Q: Does it have a FD?)

(Q: Is it in BCNF?)

(Q: Where does the redundancy come from?)

Note:
- Two independent 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 × 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 $X \rightarrow Y$**

- **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”

(Explanatio using $Y$ circle diagram)

- $X \rightarrow Y$ 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 → ta? cnum → sid?

- **COMPLEMENTATION RULE**
  - $X \rightarrow Y$, then $X \rightarrow Z$ where $Z$ is all attributes in $R$ except $X$ and $Y$
  - Proof: swapping $Y$ is the same as swapping $Z$

- **TRIVIAL MVD**
  - $X \rightarrow Y$ is trivial MVD if
    1. $Y \subseteq X$ or
    2. $X \cup Y = R$

(Prove by canonical database)

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$y_1$</td>
<td>$z_1$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$y_2$</td>
<td>$z_2$</td>
</tr>
</tbody>
</table>

→ $x_1$ | $y_1$ | $z_2$ |
FOURTH NORMAL FORM (4NF)

- Definition: R is in 4NF if for every nontrivial FD X → Y or MVD X ↠ Y, X contains a key.

- Q: Relationship between BCNF and 4NF?

\[ R(A_1, A_2, \ldots) \]
\[ X_1 \rightarrow Y_1 \]
\[ X_2 \rightarrow Y_2 \]
\[ \ldots \]
\[ Z_1 \rightarrow U_1 \]
\[ Z_2 \rightarrow U_2 \]
\[ \ldots \]

In BCNF, all \( X_i \) should contain a key.
In 4NF, all \( X_i \)'s and \( Z_i \)'s should contain a key.
(Venn 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 \( X \rightarrow Y \) 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 \( \rightarrow \) 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 \rightarrow ta

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

- Q: It is in 4NF?

- Q: Normalize into 4NF

* 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 \( X \rightarrow Y \)
  - Proof: If \( X \rightarrow Y \), swapping \( Y \) values does not create new tuples.

  \[
  \begin{array}{ccc}
  \text{X} & \text{Y} & \text{Z} \\
  \hline
  x_1 & y_1 & z_1 \\
  x_1 & y_2 & z_2 \\
  \rightarrow x_1 & y_1 & z_2 \\
  \end{array}
  \]

  (Prove by canonical database)

  - Simplified definition of 4NF: \( R \) is in 4NF if for every nontrivial MVD \( X \rightarrow Y \), \( X \) contains a key
– Since \( X \rightarrow Y \) implies \( X \rightarrow Y \), 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 \( \rightarrow \) office, office \( \rightarrow \) (phone, fax)

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