Announcement

• Project 1 Milestone 2 due tonight!

• Read the textbook!
  – Chapter 8:
    • Will cover later; But self-study the chapter
  – Chapter 14:
    • Section 14.1 – 14.5
  – Chapter 15:
    • Section 15.1 – 15.4
Student Feedback

  ➔ Feedback Form
My schedule doesn't fit with any of the times. A Friday afternoon would be great. Or Tuesday, Thursday afternoon. Or can I just ask my questions and ask what happened at workshops during office hours?

Yes. That’s fine.

Workshops are meant to help you better.

We know cases where students are too shy to come to office hours. In a group setting they usually perform better. Also, they can work as a team solving Problem Set problems.
Regarding today's (Feb. 19) announcement, I don't think laptops should be banned from class. I use my laptop as my primary means of taking notes, since I am much more organized digitally. My laptop is my only note-taking tool for all my classes and I find it much more useful than a normal notebook. 

If the reason that laptops are banned is that they distract other students, then make a rule that people using laptops sit in the back, while those taking notes sit in the front. This gives people the freedom they should have in class without hurting other students. 

If any of you have this same issue, please contact me after the class.

Read this post:

Agenda

1. 2NF, 3NF and Boyce-Codd Normal Form

2. Decompositions
Functional Dependencies (Graphical Representation)

(a) EMP_DEPT

(b) EMP_PROJ
Prime and Non-prime attributes

- A **Prime** attribute must be a member of some candidate key.

- A **Nonprime** attribute is not a prime attribute—that is, it is not a member of any candidate key.
Now that we know how to find FDs, it’s a straight-forward process:

1. Search for “bad” FDs

2. If there are any, then keep decomposing the table into sub-tables until no more bad FDs

3. When done, the database schema is normalized
Main idea is that we define “good” and “bad” FDs as follows:

- \( X \rightarrow A \) is a “good FD” if \( X \) is a (super)key
  - In other words, if \( A \) is the set of all attributes

- \( X \rightarrow A \) is a “bad FD” otherwise

We will try to eliminate the “bad” FDs!

- Via normalization
Second Normal Form

• Uses the concepts of **FDs, primary key**

• Definitions
  
  – **Full functional dependency:**
    
    • a FD \( Y \rightarrow Z \) where removal of any attribute from \( Y \) means the FD does not hold any more
Second Normal Form (cont.)

• Examples:
  – \{Ssn, Pnumber\} \rightarrow\! \text{Hours} is a full FD since neither
    • Ssn \rightarrow \text{Hours} nor Pnumber \rightarrow \text{Hours} hold
  – \{Ssn, Pnumber\} \rightarrow\! \text{Ename} is not a full FD (it is called a partial dependency) since Ssn \rightarrow \text{Ename} also holds
Second Normal Form (2)

• A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key

• R which is not in 2NF can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”
Normalizing into 2NF

(a)

EMP_PROJ

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Pnumber</th>
<th>Hours</th>
<th>Ename</th>
<th>Pname</th>
<th>Plocation</th>
</tr>
</thead>
</table>

FD1
FD2
FD3

2NF Normalization

EP1

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Pnumber</th>
<th>Hours</th>
</tr>
</thead>
</table>

FD1

EP2

<table>
<thead>
<tr>
<th>Ssn</th>
<th>Ename</th>
</tr>
</thead>
</table>

FD2

EP3

<table>
<thead>
<tr>
<th>Pnumber</th>
<th>Pname</th>
<th>Plocation</th>
</tr>
</thead>
</table>

FD3
• Definition:
  – **Transitive functional dependency:**
    • a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$

• Examples:
  – $Ssn \rightarrow Dmgr\_ssn$ is a **transitive** FD
    • Since $Ssn \rightarrow Dnumber$ and $Dnumber \rightarrow Dmgr\_ssn$ hold
  – $Ssn \rightarrow Ename$ is **non-transitive**
    • Since there is no set of attributes $X$ where $Ssn \rightarrow X$ and $X \rightarrow Ename$
Third Normal Form (2)

- A relation schema $R$ is in **third normal form (3NF)** if it is in 2NF and no non-prime attribute $A$ in $R$ is transitively dependent on the primary key.
- $R$ can be decomposed into 3NF relations via the process of 3NF normalization.
Normalizing into 3NF

(b) EMP_DEPT

<table>
<thead>
<tr>
<th>Ename</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
</table>

3NF Normalization

ED1

<table>
<thead>
<tr>
<th>Ename</th>
<th>Ssn</th>
<th>Bdate</th>
<th>Address</th>
<th>Dnumber</th>
</tr>
</thead>
</table>

ED2

<table>
<thead>
<tr>
<th>Dnumber</th>
<th>Dname</th>
<th>Dmgr_ssn</th>
</tr>
</thead>
</table>
CREATE TABLE Employee_Mgr(
  ssn char(11),
  name varchar(30),
  lot integer,
  since date,
  did integer,
  PRIMARY KEY (did),
  FOREIGN KEY (did) REFERENCES Departments (did)
  ON DELETE RESTRICT,
)
Answer: Is it in 2NF?

• Yes.

• Why?

• Every Non prime attribute is fully functionally dependent on primary key {did}
No.

Why?

Non-prime attributes name, lot are transitively dependent on did

did $\rightarrow$ ssn $\rightarrow$ \{name,lot\}
Remedy

• Take a new relation with \{ssn, name, lot\}
Remedy (Normalization)

• How to perform 1NF normalization?
  – Form new relations for each multivalued attribute

• How to perform 2NF normalization?
  – Decompose and set up a new relation for each partial key with it’s dependent attributes.

• How to perform 3NF normalization?
  – Decompose and set up a new relation that includes the non-key attribute and every-other non-key attributes it determines.
Normal Forms Defined Informally

• 1\textsuperscript{st} normal form
  – All attributes depend on the key

• 2\textsuperscript{nd} normal form
  – All attributes depend on the whole key

• 3\textsuperscript{rd} normal form
  – All attributes depend on nothing but the key
A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on **every** key of R

A relation schema R is in **third normal form (3NF)** if it is in 2NF and **no non-prime attribute A** in R is transitively dependent on **any** key of R
Normalization into 2NF

(a)

Let's consider a table called `LOTS` with the following columns:
- Property_id#
- County_name
- Lot#
- Area
- Price
- Tax_rate

Candidate Key

FD1
FD2
FD3
FD4

(b)

Now, consider two new tables:

- `LOTS1` with columns:
  - Property_id#
  - County_name
  - Lot#
  - Area
  - Price

FD1
FD2
FD4

- `LOTS2` with columns:
  - County_name
  - Tax_rate

FD3
Normalization into 3NF

(b) LOTS1

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) LOTS1A

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LOTS1B

<table>
<thead>
<tr>
<th>Area</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD4</td>
<td></td>
</tr>
</tbody>
</table>
1. BOYCE-CODD NORMAL FORM
What you will learn about in this section

1. Conceptual Design
2. Boyce-Codd Normal Form
3. The BCNF Decomposition Algorithm
BCNF (Boyce-Codd Normal Form)

• A relation schema \( R \) is in Boyce-Codd Normal Form (BCNF) if whenever an \( \text{FD} \ X \rightarrow A \) holds in \( R \), then \( X \) is a superkey of \( R \).

• Each normal form is strictly stronger than the previous one
  – Every 2NF relation is in 1NF
  – Every 3NF relation is in 2NF
  – Every BCNF relation is in 3NF
General Definition of 2NF and 3NF

• A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on **every** key of R.

• A relation schema R is in **third normal form (3NF)** if it is in 2NF and no non-prime attribute A in R is transitively dependent on **any** key of R.

**wait a sec... If A is a prime attribute, it’s still in 3NF?!**

**Yes! That’s correct**
Which Normal form is this?

<table>
<thead>
<tr>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

FD1 [_____]            
FD2 [_____]            

A relation schema R is in **third normal form (3NF)** if it is in 2NF and no non-prime attribute A in R is transitively dependent on **any** key of R.

It is in 3NF, but not in BCNF because C → B.
Boyce-Codd normal form

(a) LOTS1A

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
</tr>
</thead>
</table>

FD1

FD2

FD5

BCNF Normalization

LOTS1AX

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>Area</th>
<th>Lot#</th>
</tr>
</thead>
</table>

LOTS1AY

<table>
<thead>
<tr>
<th>Area</th>
<th>County_name</th>
</tr>
</thead>
</table>
ALTERNATIVE DEFINITION of 3NF: We can restate the definition as:

A relation schema $R$ is in **third normal form (3NF)** if, whenever a nontrivial FD $X \rightarrow A$ holds in $R$, either

a) $X$ is a superkey of $R$ or

b) $A$ is a prime attribute of $R$

The condition (b) takes care of the dependencies that “slip through” (are allowable to) 3NF but are “caught by” BCNF which we discuss next.
BCNF (Boyce-Codd Normal Form)

• Definition of 3NF:
  A relation schema R is in **3NF** if, whenever a nontrivial FD $X \rightarrow A$ holds in R, either
  a) $X$ is a superkey of R or
  b) $A$ is a prime attribute of R

• A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an FD $X \rightarrow A$ holds in R, then
  a) $X$ is a superkey of R
  b) There is no b

• Each normal form is strictly stronger than the previous one
  – Every 2NF relation is in 1NF
  – Every 3NF relation is in 2NF
  – Every BCNF relation is in 3NF
Boyce-Codd normal form

(a) LOTS1A

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>County_name</th>
<th>Lot#</th>
<th>Area</th>
</tr>
</thead>
</table>

FD1

FD2

FD5

BCNF Normalization

LOTS1AX

<table>
<thead>
<tr>
<th>Property_id#</th>
<th>Area</th>
<th>Lot#</th>
</tr>
</thead>
</table>

LOTS1AY

<table>
<thead>
<tr>
<th>Area</th>
<th>County_name</th>
</tr>
</thead>
</table>

(b) $R$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
</table>

FD1

FD2

Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. $C \rightarrow B$. 
A relation TEACH that is in 3NF but not in BCNF

- Two FDs exist in the relation TEACH:
  - \{student, course\} \rightarrow instructor
  - instructor \rightarrow course

- \{student, course\} is a candidate key for this relation
- So this relation is in **3NF but not in BCNF**
- A relation **NOT** in BCNF should be decomposed
  - while possibly forgoing the preservation of all functional dependencies in the decomposed relations.

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Narayan</td>
<td>Database</td>
<td>Mark</td>
</tr>
<tr>
<td>Smith</td>
<td>Database</td>
<td>Navathe</td>
</tr>
<tr>
<td>Smith</td>
<td>Operating Systems</td>
<td>Ammar</td>
</tr>
<tr>
<td>Smith</td>
<td>Theory</td>
<td>Schulman</td>
</tr>
<tr>
<td>Wallace</td>
<td>Database</td>
<td>Mark</td>
</tr>
<tr>
<td>Wallace</td>
<td>Operating Systems</td>
<td>Ahamad</td>
</tr>
<tr>
<td>Wong</td>
<td>Database</td>
<td>Omieciniski</td>
</tr>
<tr>
<td>Zelaya</td>
<td>Database</td>
<td>Navathe</td>
</tr>
<tr>
<td>Narayan</td>
<td>Operating Systems</td>
<td>Ammar</td>
</tr>
</tbody>
</table>
Achieving the BCNF by Decomposition

• Three possible decompositions for relation TEACH
  – D1: \{student, instructor\} and \{student, course\}
  – D2: \{course, instructor\} and \{course, student\}
  ✓ D3: \{instructor, course\} and \{instructor, student\}
Boyce-Codd Normal Form

BCNF is a simple condition for removing anomalies from relations:

A relation $R$ is **in BCNF** if:

if $\{X_1, \ldots, X_n\} \rightarrow A$ is a *non-trivial* FD in $R$
then $\{X_1, \ldots, X_n\}$ is a *superkey* for $R$

In other words: there are no “bad” FDs
What is the key?

{SSN, PhoneNumber}

Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-1234</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

{SSN} $\rightarrow$ {Name, City}

This FD is bad because it is not a superkey

$\Rightarrow$ **Not** in BCNF
### Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>Madison</td>
</tr>
</tbody>
</table>

Now in BCNF!

{SSN} \(\rightarrow\) \{Name, City\}

This FD is now *good* because it is the key.

Let’s check anomalies:
- Redundancy?
- Update?
- Delete?
BCNF Decomposition Algorithm

BCNFDcomp(R):

Find X s.t. X + ≠ X and X + ≠ [all attributes]
if (not found) then Return R
let Y = X + - X, Z = (X + )C
decompose R into R1(X È Y) and R2(X È Z)
Return BCNFDcomp(R1), BCNFDcomp(R2)
BCNF Decomposition Algorithm

BCNFDcomp(R):
Find a set of attributes X s.t.: \(X^+ \neq X\) and \(X^+ \neq \) [all attributes]

Find a set of attributes X which has non-trivial “bad” FDs, i.e. is not a superkey, using closures
BCNF Decomposition Algorithm

BCNFDegomp(R):
Find a set of attributes X s.t.: X⁺ ≠ X and X⁺ ≠ [all attributes]

  if (not found) then Return R

If no “bad” FDs found, in BCNF!
BCNF Decomposition Algorithm

BCNFDcomp(R):
Find a set of attributes X s.t.: X⁺ ≠ X and X⁺ ≠ [all attributes]

if (not found) then Return R

let Y = X⁺ - X, Z = (X⁺)ᶜ

Let Y be the attributes that X functionally determines (+ that are not in X)

And let Z be the other attributes that it doesn’t
BCNFDecomp(R):
Find a set of attributes X s.t.: $X^+ \neq X$ and $X^+ \neq \{\text{all attributes}\}$

if (not found) then Return R

let $Y = X^+ - X, Z = (X^+)^C$

decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$
BCNF Decomposition Algorithm

BCNFDecomp(R):
Find a set of attributes X s.t.: $X^+ \neq X$ and $X^+ \neq [\text{all attributes}]

if (not found) then Return R

let $Y = X^+ - X, \ Z = (X^+)^C$

decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

And one relation with X plus the attributes it does not determine (Z)
BCNF Decomposition Algorithm

BCNFDegomp(R):
Find a set of attributes $X$ s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) then Return R

let $Y = X^+ - X$, $Z = (X^+)^C$
decompose $R$ into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

Return BCNFDegomp($R_1$), BCNFDegomp($R_2$)

Proceed recursively until no more “bad” FDs!
Another way of representing the same concept

BCNFDcomp(R):
If $X \rightarrow A$ causes BCNF violation:

Decompose $R$ into

$R_1 = XA$
$R_2 = R - A$

(Note: $X$ is present in both $R_1$ and $R_2$)
BCNFDecomp(R):

Find a set of attributes X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) then Return R

let $Y = X^+ - X$, $Z = (X^+)^C$

decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

Return BCNFDecomp($R_1$), BCNFDecomp($R_2$)

BCNFDecomp(R):

If $X \rightarrow A$ causes BCNF violation:

Decompose R into

$R_1 = XA$
$R_2 = R - A$

(Note: X is present in both $R_1$ and $R_2$)

Example

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

$\{A\} \rightarrow \{B,C\}$
$\{C\} \rightarrow \{D\}$
Example

\[ R(A,B,C,D,E) \]
\[ \{A\}^+ = \{A,B,C,D\} \neq \{A,B,C,D,E\} \]

\[ R_1(A,B,C,D) \]
\[ \{C\}^+ = \{C,D\} \neq \{A,B,C,D\} \]

\[ R_{11}(C,D) \]
\[ R_{12}(A,B,C) \]

\[ R_2(A,E) \]

\[ \{A\} \rightarrow \{B,C\} \]
\[ \{C\} \rightarrow \{D\} \]
Another Example

• Let the relation schema $R(A,B,C,D)$ is given. For each of the following set of FDs do the following:

• i) indicate all the BCNF violations. Do not forget to consider FD’s that are not in the given set. However, it is not required to give violations that have more than one attribute on the right side.

• ii) Decompose the relations, as necessary, into collections of relations that are in BCNF.

• 1. FDs $AB \rightarrow C$, $C \rightarrow D$, and $D \rightarrow A$
Find Non-trivial dependencies

There are 14 nontrivial dependencies. They are:
C→A, C→D, D→A, AB→D, AB→C,
AC→D, BC→A, BC→D, BD→A, BD→C, CD→A,
ABC→D, ABD→C, and BCD→A.
• One choice is to decompose using the violation C→D.
• Using the above FDs, we get ACD (Because of C→D and C→A) and BC as decomposed relations.
• BC is surely in BCNF, since any two-attribute relation is.
• we discover that ACD is not in BCNF since C is its only key.
• We must further decompose ACD into AD and CD.
• Thus, the three relations of the decomposition are BC, AD, and CD.
Two other topics to Study

• Cover and Minimal Cover

• Let $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$.
• Let $G = \{A \rightarrow CD, E \rightarrow AH\}$.

• Show that:
• 1. $G$ covers $F$
• 2. $F$ covers $G$
• 3. $F$ and $G$ are equivalent
Cover

• We say that a set of functional dependencies $F$ covers another set of functional dependencies $G$,

  – if every functional dependency in $G$ can be inferred from $F$.

  – More formally, $F$ covers $G$ if $G^+ \subseteq F^+$

  – $F$ is a minimal cover of $G$ if $F$ is the smallest set of functional dependencies that cover $G$
1. Show: G covers F or \((F \subseteq G^+)\)

let’s check each FD in \(F\):

- \(A \rightarrow C\)
  So, let’s find \(A^+\) in \(G\).
  \(A^+\) in \(G = \{ACD\}\) which includes \(C\). So, continue;

- \(AC \rightarrow D\)
  Let’s get \(AC^+\) in \(G\),
  \(AC^+\) in \(G = \{ACD\}\) which contains \(D\). So, continue;

- \(E \rightarrow AD\)
  Let’s get \(E^+\) in \(G\),
  \(E^+\) in \(G = \{EACDH\}\) which contains \(AD\). So, continue;

We found that every dependency in \(F\) can be inferred from \(G\).
So, we can say that \(G\) covers \(F\).
Acknowledgement

• Some of the slides in this presentation are taken from the slides provided by the authors.
• Many of these slides are taken from cs145 course offered by Stanford University.
• Thanks to YouTube, especially to Dr. Daniel Soper for his useful videos.