Lecture 05 : Functional Dependencies (FDs)

Outline
- ER is for top-down and informal (but rigorous) design
- FDs are used for bottom-up and formal design and analysis
- Update anomalies
- Reasoning about Functional Dependencies
- Heath's rule
## Update anomalies

### Big Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>college</th>
<th>course</th>
<th>part</th>
<th>term_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy88</td>
<td>Yoni</td>
<td>New Hall</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>uu99</td>
<td>Uri</td>
<td>King’s</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
</tbody>
</table>

- How can we tell if an insert record is consistent with current records?
- Can we record data about a course before students enroll?
- Will we wipe out information about a college when last student associated with the college is deleted?

---

**Redundancy implies more locking ...**

... at least for correct transactions!

### Big Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>college</th>
<th>course</th>
<th>part</th>
<th>term_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy88</td>
<td>Yoni</td>
<td>New Hall</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>uu99</td>
<td>Uri</td>
<td>King’s</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
</tbody>
</table>

- Change **New Hall** to **Murray Edwards College**
  - Conceptually simple update
  - May require locking entire table.
Redundancy is the root of (almost) all database evils

- It may not be obvious, but redundancy is also the cause of update anomalies.
- By redundancy we do not mean that some values occur many times in the database!
  - A foreign key value may be have millions of copies!
- But then, what do we mean?

Functional Dependency

### Functional Dependency (FD)

Let $R(X)$ be a relational schema and $Y \subseteq X$, $Z \subseteq X$ be two attribute sets. We say $Y$ functionally determines $Z$, written $Y \rightarrow Z$, if for any two tuples $u$ and $v$ in an instance of $R(X)$ we have

$$u.Y = v.Y \rightarrow u.Z = v.Z.$$  

We call $Y \rightarrow Z$ a functional dependency.

A functional dependency is a semantic assertion. It represents a rule that should always hold in any instance of schema $R(X)$. 
Example FDs

Big Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>college</th>
<th>course</th>
<th>part</th>
<th>term_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy88</td>
<td>Yoni</td>
<td>New Hall</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>uu99</td>
<td>Uri</td>
<td>King’s</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
</tbody>
</table>

- sid → name
- sid → college
- course → part
- course → term_name

Keys, revisited

Candidate Key
Let $R(X)$ be a relational schema and $Y \subseteq X$. $Y$ is a candidate key if
- The FD $Y \rightarrow X$ holds, and
- for no proper subset $Z \subset Y$ does $Z \rightarrow X$ hold.

Prime and Non-prime attributes
An attribute $A$ is prime for $R(X)$ if it is a member of some candidate key for $R$. Otherwise, $A$ is non-prime.

Database redundancy roughly means the existence of non-key functional dependencies!
Closure

By soundness and completeness

\[
\text{closure}(F, X) = \{ A \mid F \vdash X \rightarrow A \} = \{ A \mid X \rightarrow A \in F^+ \}
\]

Claim 2 (from previous lecture)

\[ Y \rightarrow W \in F^+ \text{ if and only if } W \subseteq \text{closure}(F, Y). \]

If we had an algorithm for closure\((F, X)\), then we would have a (brute force!) algorithm for enumerating \(F^+\):

\[ F^+ \]

- for every subset \(Y \subseteq \text{atts}(F)\)
  - for every subset \(Z \subseteq \text{closure}(F, Y)\),
    - output \(Y \rightarrow Z\)

Attribute Closure Algorithm

- Input: a set of FDs \(F\) and a set of attributes \(X\).
- Output: \(Y = \text{closure}(F, X)\)

1. \(Y := X\)
2. while there is some \(S \rightarrow T \in F\) with \(S \subseteq Y\) and \(T \not\subseteq Y\), then \(Y := Y \cup T\).
An Example (UW1997, Exercise 3.6.1)

$R(A, B, C, D)$ with $F$ made up of the FDs

$$
A, B \rightarrow C \\
C \rightarrow D \\
D \rightarrow A
$$

What is $F^+$?

Brute force!

Let’s just consider all possible nonempty sets $X$ — there are only 15...

Example (cont.)

$$F = \{A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A\}$$

For the single attributes we have

- $\{A\}^+ = \{A\}$,
- $\{B\}^+ = \{B\}$,
- $\{C\}^+ = \{A, C, D\}$,
  - $\{C\} \xrightarrow{C \rightarrow D} \{C, D\} \xrightarrow{D \rightarrow A} \{A, C, D\}$
- $\{D\}^+ = \{A, D\}$
  - $\{D\} \xrightarrow{D \rightarrow A} \{A, D\}$

The only new dependency we get with a single attribute on the left is $C \rightarrow A$. 
Example (cont.)

\[ F = \{ A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A \} \]

Now consider pairs of attributes.

- \( \{ A, B \}^+ = \{ A, B, C, D \} \),
  - so \( A, B \rightarrow D \) is a new dependency
- \( \{ A, C \}^+ = \{ A, C, D \} \),
  - so \( A, C \rightarrow D \) is a new dependency
- \( \{ A, D \}^+ = \{ A, D \} \),
  - so nothing new.
- \( \{ B, C \}^+ = \{ A, B, C, D \} \),
  - so \( B, C \rightarrow A, D \) is a new dependency
- \( \{ B, D \}^+ = \{ A, B, C, D \} \),
  - so \( B, D \rightarrow A, C \) is a new dependency
- \( \{ C, D \}^+ = \{ A, C, D \} \),
  - so \( C, D \rightarrow A \) is a new dependency

Example (cont.)

\[ F = \{ A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A \} \]

For the triples of attributes:

- \( \{ A, C, D \}^+ = \{ A, C, D \} \),
- \( \{ A, B, D \}^+ = \{ A, B, C, D \} \),
  - so \( A, B, D \rightarrow C \) is a new dependency
- \( \{ A, B, C \}^+ = \{ A, B, C, D \} \),
  - so \( A, B, C \rightarrow D \) is a new dependency
- \( \{ B, C, D \}^+ = \{ A, B, C, D \} \),
  - so \( B, C, D \rightarrow A \) is a new dependency

And since \( \{ A, B, C, D \}^+ = \{ A, B, C, D \} \), we get no new dependencies with four attributes.
We generated 11 new FDs:

\[
\begin{align*}
    C & \rightarrow A & A, B & \rightarrow D \\
    A, C & \rightarrow D & B, C & \rightarrow A \\
    B, C & \rightarrow D & B, D & \rightarrow A \\
    B, D & \rightarrow C & C, D & \rightarrow A \\
    A, B, C & \rightarrow D & A, B, D & \rightarrow C \\
    B, C, D & \rightarrow A
\end{align*}
\]

**Can you see the Key?**

\{A, B\}, \{B, C\}, and \{B, D\} are keys.

**Note:** this schema is already in 3NF! Why?

**Semantic Closure**

**Notation**

\[ F \models Y \rightarrow Z \]

means that any database instance that that satisfies every FD of \( F \), must also satisfy \( Y \rightarrow Z \).

The **semantic closure** of \( F \), denoted \( F^+ \), is defined to be

\[
F^+ = \{ Y \rightarrow Z \mid Y \cup Z \subseteq \text{atts}(F) \text{ and } \models F \rightarrow Y \rightarrow Z \}.
\]

The **membership problem** is to determine if \( Y \rightarrow Z \in F^+ \).
Reasoning about Functional Dependencies

We write $F \vdash Y \rightarrow Z$ when $Y \rightarrow Z$ can be derived from $F$ via the following rules.

**Armstrong’s Axioms**

- **Reflexivity** If $Z \subseteq Y$, then $F \vdash Y \rightarrow Z$.
- **Augmentation** If $F \vdash Y \rightarrow Z$ then $F \vdash Y, W \rightarrow Z, W$.
- **Transitivity** If $F \vdash Y \rightarrow Z$ and $F \models Z \rightarrow W$, then $F \vdash Y \rightarrow W$.

Logical Closure (of a set of attributes)

**Notation**

$$\text{closure}(F, X) = \{ A \mid F \vdash X \rightarrow A \}$$

**Claim 1**

If $Y \rightarrow W \in F$ and $Y \subseteq \text{closure}(F, X)$, then $W \subseteq \text{closure}(F, X)$.

**Claim 2**

$Y \rightarrow W \in F^+$ if and only if $W \subseteq \text{closure}(F, Y)$.
Soundness and Completeness

Soundness

\[ F \vdash f \iff f \in F^+ \]

Completeness

\[ f \in F^+ \implies F \vdash f \]

Proof of Completeness (soundness left as an exercise)

Show \(- (F \vdash f) \implies \neg (F \models f)\):

- Suppose \(- (F \vdash Y \rightarrow Z)\) for \(R(X)\).
- Let \(Y^+ = \text{closure}(F, Y)\).
- \(\exists B \in Z, \text{ with } B \notin Y^+\).
- Construct an instance of \(R\) with just two records, \(u\) and \(v\), that agree on \(Y^+\) but not on \(X - Y^+\).
- By construction, this instance does not satisfy \(Y \rightarrow Z\).
- But it does satisfy \(F\)! Why?
  - Let \(S \rightarrow T\) be any FD in \(F\), with \(u.S = v.S\).
  - So \(S \subseteq Y^+\), and so \(T \subseteq Y^+\) by claim 1,
  - and so \(u.T = v.T\)
Consequences of Armstrong’s Axioms

Union  If \( F \models Y \rightarrow Z \) and \( F \models Y \rightarrow W \), then \( F \models Y \rightarrow W, Z \).

Pseudo-transitivity  If \( F \models Y \rightarrow Z \) and \( F \models U, Z \rightarrow W \), then 
\( F \models Y, U \rightarrow W \).

Decomposition  If \( F \models Y \rightarrow Z \) and \( W \subseteq Z \), then \( F \models Y \rightarrow W \).

Exercise: Prove these using Armstrong’s axioms!

Proof of the Union Rule

Suppose we have
\[
F \models Y \rightarrow Z,
F \models Y \rightarrow W.
\]

By augmentation we have
\[
F \models Y, Y \rightarrow Y, Z,
\]
that is,
\[
F \models Y \rightarrow Y, Z.
\]

Also using augmentation we obtain
\[
F \models Y, Z \rightarrow W, Z.
\]

Therefore, by transitivity we obtain
\[
F \models Y \rightarrow W, Z.
\]
Example application of functional reasoning.

**Heath’s Rule**

Suppose $R(A, B, C)$ is a relational schema with functional dependency $A \rightarrow B$, then

$$R = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R).$$

**Proof of Heath’s Rule**

We first show that $R \subseteq \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R)$.

- If $u = (a, b, c) \in R$, then $u_1 = (a, b) \in \pi_{A,B}(R)$ and $u_2 = (a, c) \in \pi_{A,C}(R)$.
- Since $\{(a, b)\} \bowtie_A \{(a, c)\} = \{(a, b, c)\}$ we know $u \in \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R)$.

In the other direction we must show $R' = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R) \subseteq R$.

- If $u = (a, b, c) \in R'$, then there must exist tuples $u_1 = (a, b) \in \pi_{A,B}(R)$ and $u_2 = (a, c) \in \pi_{A,C}(R)$.
- This means that there must exist a $u' = (a, b', c) \in R$ such that $u_2 = \pi_{A,C}((a, b', c))$.
- However, the functional dependency tells us that $b = b'$, so $u = (a, b, c) \in R$. 


Closure Example

$R(A, B, C, D, D, F)$ with

- $A, B \rightarrow C$
- $B, C \rightarrow D$
- $D \rightarrow E$
- $C, F \rightarrow B$

What is the closure of $\{A, B\}$?

$\{A, B\}$

$\xrightarrow{AB \rightarrow C} \{A, B, C\}$

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

$\xrightarrow{D \rightarrow E} \{A, B, C, D, E\}$

So $\{A, B\}^+ = \{A, B, C, D, E\}$ and $A, B \rightarrow C, D, E$.

Lecture 06 : Normal Forms

Outline

- First Normal Form (1NF)
- Second Normal Form (2NF)
- 3NF and BCNF
- Multi-valued dependencies (MVDs)
- Fourth Normal Form
First Normal Form (1NF)

We will assume every schema is in 1NF.

1NF

A schema \( R(A_1 : S_1, A_2 : S_2, \ldots, A_n : S_n) \) is in First Normal Form (1NF) if the domains \( S_i \) are elementary — their values are atomic.

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timothy George Griffin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>first_name</th>
<th>middle_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timothy</td>
<td>George</td>
<td>Griffin</td>
</tr>
</tbody>
</table>

Second Normal Form (2NF)

Second Normal Form (2CNF)

A relational schema \( R \) is in 2NF if for every functional dependency \( X \rightarrow A \) either
- \( A \in X \), or
- \( X \) is a superkey for \( R \), or
- \( A \) is a member of some key, or
- \( X \) is not a proper subset of any key.
3NF and BCNF

Third Normal Form (3CNF)
A relational schema \( R \) is in 3NF if for every functional dependency \( X \rightarrow A \) either
- \( A \in X \), or
- \( X \) is a superkey for \( R \), or
- \( A \) is a member of some key.

Boyce-Codd Normal Form (BCNF)
A relational schema \( R \) is in BCNF if for every functional dependency \( X \rightarrow A \) either
- \( A \in X \), or
- \( X \) is a superkey for \( R \).

Is something missing?

Another look at Heath’s Rule

Given \( R(Z, W, Y) \) with FDs \( F \)
If \( Z \rightarrow W \in F^+ \), the
\[
R = \pi_{Z,W}(R) \bowtie \pi_{Z,Y}(R)
\]

What about an implication in the other direction? That is, suppose we have
\[
R = \pi_{Z,W}(R) \bowtie \pi_{Z,Y}(R).
\]

Q Can we conclude anything about FDs on \( R \)? In particular, is it true that \( Z \rightarrow W \) holds?
A No!
We just need one counter example ...

\[ R = \pi_{A,B}(R) \times \pi_{A,C}(R) \]

\[
\begin{array}{ccc}
A & B & C \\
\hline
a & b_1 & c_1 \\
a & b_2 & c_2 \\
a & b_1 & c_2 \\
a & b_2 & c_1 \\
\end{array}
\quad
\begin{array}{ccc}
A & B \\
\hline
a & b_1 \\
a & b_2 \\
a & a \\
\end{array}
\quad
\begin{array}{ccc}
A & C \\
\hline
a & c_1 \\
a & c_2 \\
a & a \\
\end{array}
\]

Clearly \( A \rightarrow B \) is not an FD of \( R \).

A concrete example

<table>
<thead>
<tr>
<th>course_name</th>
<th>lecturer</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>Tim</td>
<td>Ullman and Widom</td>
</tr>
<tr>
<td>Databases</td>
<td>Fatima</td>
<td>Date</td>
</tr>
<tr>
<td>Databases</td>
<td>Tim</td>
<td>Date</td>
</tr>
<tr>
<td>Databases</td>
<td>Fatima</td>
<td>Ullman and Widom</td>
</tr>
</tbody>
</table>

Assuming that texts and lecturers are assigned to courses independently, then a better representation would in two tables:

<table>
<thead>
<tr>
<th>course_name</th>
<th>lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>Tim</td>
</tr>
<tr>
<td>Databases</td>
<td>Fatima</td>
</tr>
<tr>
<td>course_name</td>
<td>text</td>
</tr>
<tr>
<td>Databases</td>
<td></td>
</tr>
<tr>
<td>Databases</td>
<td>Ullman and Widom</td>
</tr>
<tr>
<td>Databases</td>
<td>Date</td>
</tr>
</tbody>
</table>
Time for a definition! MVDs

**Multivalued Dependencies (MVDs)**

Let \( R(Z, W, Y) \) be a relational schema. A multivalued dependency, denoted \( Z \rightarrow W \), holds if whenever \( t \) and \( u \) are two records that agree on the attributes of \( Z \), then there must be some tuple \( \nu \) such that

1. \( \nu \) agrees with both \( t \) and \( u \) on the attributes of \( Z \),
2. \( \nu \) agrees with \( t \) on the attributes of \( W \),
3. \( \nu \) agrees with \( u \) on the attributes of \( Y \).

---

**A few observations**

**Note 1**

Every functional dependency is multivalued dependency,

\[
(Z \rightarrow W) \implies (Z \rightarrow W).
\]

To see this, just let \( \nu = u \) in the above definition.

**Note 2**

Let \( R(Z, W, Y) \) be a relational schema, then

\[
(Z \rightarrow W) \iff (Z \rightarrow Y),
\]

by symmetry of the definition.
MVDs and lossless-join decompositions

Fun Fun Fact
Let \( R(Z, W, Y) \) be a relational schema. The decomposition \( R_1(Z, W), R_2(Z, Y) \) is a lossless-join decomposition of \( R \) if and only if the MVD \( Z \rightarrow W \) holds.

Proof of Fun Fun Fact

Proof of \( (Z \rightarrow W) \iff R = \pi_{Z,W}(R) \bowtie \pi_{Z,Y}(R) \)

- Suppose \( Z \rightarrow W \).
- We know (from proof of Heath’s rule) that \( R \subseteq \pi_{Z,W}(R) \bowtie \pi_{Z,Y}(R) \).
- So we only need to show \( \pi_{Z,W}(R) \bowtie \pi_{Z,Y}(R) \subseteq R \).
- Suppose \( r \in \pi_{Z,W}(R) \bowtie \pi_{Z,Y}(R) \).
- So there must be a \( t \in R \) and \( u \in R \) with
  \[ \{r\} = \pi_{Z,W}([t]) \bowtie \pi_{Z,Y}([u]). \]
- In other words, there must be a \( t \in R \) and \( u \in R \) with \( t.Z = u.Z \).
- So the MVD tells us that then there must be some tuple \( v \in R \) such that
  - \( v \) agrees with both \( t \) and \( u \) on the attributes of \( Z \),
  - \( v \) agrees with \( t \) on the attributes of \( W \),
  - \( v \) agrees with \( u \) on the attributes of \( Y \).
- This \( v \) must be the same as \( r \), so \( r \in R \).
Proof of Fun Fun Fact (cont.)

Proof of $R = \pi_{Z,W}(R) \times \pi_{Z,Y}(R) \rightarrowtail (Z \rightarrow W)$

- Suppose $R = \pi_{Z,W}(R) \times \pi_{Z,Y}(R)$.
- Let $t$ and $u$ be any records in $R$ with $t.Z = u.Z$.
- Let $v$ be defined by $\{v\} = \pi_{Z,W}(\{t\}) \times \pi_{Z,Y}(\{u\})$ (and we know $v \in R$ by the assumption).
- Note that by construction we have:
  1. $v.Z = t.Z = u.Z$,
  2. $v.W = t.W$,
- Therefore, $Z \rightarrow W$ holds.

Fourth Normal Form

Trivial MVD

The MVD $Z \rightarrow W$ is **trivial** for relational schema $R(Z, W, Y)$ if

- $Z \cap W \neq \{\}$, or
- $Y = \{\}$.

4NF

A relational schema $R(Z, W, Y)$ is in 4NF if for every MVD $Z \rightarrow W$ either

- $Z \rightarrow W$ is a trivial MVD, or
- $Z$ is a superkey for $R$.

Note: $4NF \subset BCNF \subset 3NF \subset 2NF$
We always want the lossless-join property. What are our options?

<table>
<thead>
<tr>
<th></th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserves FDs</td>
<td>Yes</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Preserves MVDs</td>
<td>Maybe</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Eliminates FD-redundancy</td>
<td>Maybe</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates MVD-redundancy</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Inclusions

Clearly BCNF $\subseteq$ 3NF $\subseteq$ 2NF. These are proper inclusions:

In 2NF, but not 3NF

$R(A, B, C)$, with $F = \{A \rightarrow B, B \rightarrow C\}$.

In 3NF, but not BCNF

$R(A, B, C)$, with $F = \{A, B \rightarrow C, C \rightarrow B\}$.

- This is in 3NF since $AB$ and $AC$ are keys, so there are no non-prime attributes
- But not in BCNF since $C$ is not a key and we have $C \rightarrow B$. 


The Plan

Given a relational schema $R(X)$ with FDs $F$:

- Reason about FDs
  - Is $F$ missing FDs that are logically implied by those in $F$?
- Decompose each $R(X)$ into smaller $R_1(X_1)$, $R_2(X_2)$, \ldots $R_k(X_k)$, where each $R_i(X_i)$ is in the desired Normal Form.

Are some decompositions better than others?

Desired properties of any decomposition

Lossless-join decomposition

A decomposition of schema $R(X)$ to $S(Y \cup Z)$ and $T(Y \cup (X - Z))$ is a lossless-join decomposition if for every database instances we have $R = S \times T$.

Dependency preserving decomposition

A decomposition of schema $R(X)$ to $S(Y \cup Z)$ and $T(Y \cup (X - Z))$ is dependency preserving, if enforcing FDs on $S$ and $T$ individually has the same effect as enforcing all FDs on $S \times T$.

We will see that it is not always possible to achieve both of these goals.
Lecture 07: Schema Decomposition

Outline
- General Decomposition Method (GDM)
- The lossless-join condition is guaranteed by GDM
- The GDM does not always preserve dependencies!

General Decomposition Method (GDM)

GDM
1. Understand your FDs $F$ (compute $F^+$),
2. find $R(X) = R(Z \cup W \cup Y)$ (sets $Z$, $W$ and $Y$ are disjoint) with FD $Z \rightarrow W \in F^+$ violating a condition of desired NF,
3. split $R$ into two tables $R_1(Z, W)$ and $R_2(Z, Y)$
4. wash, rinse, repeat

Reminder
For $Z \rightarrow W$, if we assume $Z \cap W = \emptyset$, then the conditions are
1. $Z$ is a superkey for $R$ (2NF, 3NF, BCNF)
2. $W$ is a subset of some key (2NF, 3NF)
3. $Z$ is not a proper subset of any key (2NF)
The lossless-join condition is guaranteed by GDM

- This method will produce a lossless-join decomposition because of (repeated applications of) Heath’s Rule!
- That is, each time we replace an $S$ by $S_1$ and $S_2$, we will always be able to recover $S$ as $S_1 \bowtie S_2$.
- Note that in GDM step 3, the FD $Z \rightarrow W$ may represent a key constraint for $R_1$.

But does the method always terminate? Please think about this....

---

General Decomposition Method Revisited

**GDM++**

- Understand your FDs and MVDs $F$ (compute $F^+$),
- find $R(X) = R(Z, W, Y)$ (sets $Z$, $W$ and $Y$ are disjoint) with either FD $Z \rightarrow W \in F^+$ or MVD $Z \rightarrow W \in F^+$ violating a condition of desired NF,
- split $R$ into two tables $R_1(Z, W)$ and $R_2(Z, Y)$
- wash, rinse, repeat
Return to Example — Decompose to BCNF

\[ R(A, B, C, D) \]

\[ F = \{ A, B \rightarrow C, \ C \rightarrow D, \ D \rightarrow A \} \]

Which FDs in \( F^+ \) violate BCNF?

\[
\begin{align*}
C & \rightarrow A \\
C & \rightarrow D \\
D & \rightarrow A \\
A, C & \rightarrow D \\
C, D & \rightarrow A
\end{align*}
\]

Decompose \( R(A, B, C, D) \) to BCNF

Use \( C \rightarrow D \) to obtain

\( R_1(C, D) \). This is in BCNF. Done.

\( R_2(A, B, C) \) This is not in BCNF. Why? \( A, B \) and \( B, C \) are the only keys, and \( C \rightarrow A \) is a FD for \( R_1 \). So use \( C \rightarrow A \) to obtain

\( R_{2.1}(A, C) \). This is in BCNF. Done.

\( R_{2.2}(B, C) \). This is in BCNF. Done.

Exercise: Try starting with any of the other BCNF violations and see where you end up.
The GDM does not always preserve dependencies!

\[
R(A, B, C, D, E)
\]

\[
\begin{align*}
A, B & \rightarrow C \\
D, E & \rightarrow C \\
B & \rightarrow D 
\end{align*}
\]

- \(\{A, B\}^+ = \{A, B, C, D\}\),
- so \(A, B \rightarrow C, D\),
- and \(\{A, B, E\}\) is a key.

- \(\{B, E\}^+ = \{B, C, D, E\}\),
- so \(B, E \rightarrow C, D\),
- and \(\{A, B, E\}\) is a key (again)

Let’s try for a BCNF decomposition ...

**Decomposition 1**

Decompose \(R(A, B, C, D, E)\) using \(A, B \rightarrow C, D\):

- \(R_1(A, B, C, D)\). Decompose this using \(B \rightarrow D\):
  - \(R_{1,1}(B, D)\). Done.
  - \(R_{1,2}(A, B, C)\). Done.
- \(R_2(A, B, E)\). Done.

But in this decomposition, how will we enforce this dependency?

\[D, E \rightarrow C\]
Decomposition 2

Decompose $R(A, B, C, D, E)$ using $B, E \rightarrow C, D$:

- $R_3(B, C, D, E)$. Decompose this using $D, E \rightarrow C$
  - $R_{3.1}(C, D, E)$. Done.
  - $R_{3.2}(B, D, E)$. Decompose this using $B \rightarrow D$:
    - $R_{3.2.1}(B, D)$. Done.
    - $R_{3.2.2}(B, E)$. Done.
- $R_4(A, B, E)$. Done.

But in this decomposition, how will we enforce this dependency?

$$A, B \rightarrow C$$

Summary

- It always is possible to obtain BCNF that has the lossless-join property (using GDM)
  - But the result may not preserve all dependencies.
- It is always possible to obtain 3NF that preserves dependencies and has the lossless-join property.
  - Using methods based on “minimal covers” (for example, see EN2000).