Keys

- A superkey is a set of attributes $A_1, \ldots, A_n$ s.t. for any other attribute $B$, we have $A_1, \ldots, A_n \rightarrow B$

- A key is a minimal superkey
  - I.e. set of attributes which is a superkey and for which no subset is a superkey
Computing (Super)Keys

- Compute $X^+$ for all sets $X$
- If $X^+ = \text{all attributes}$, then $X$ is a superkey
- List only the minimal $X$’s to get the keys
Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key?
Example

Product(name, price, category, color)

name, category → price

Hence (name, category) is a key
Examples of Keys

Enrollment(student, address, course, room, time)

- student $\rightarrow$ address
- room, time $\rightarrow$ course
- student, course $\rightarrow$ room, time

Find the keys
Eliminating Anomalies

Main idea:

- $X \rightarrow A$ is OK if $X$ is a (super)key
- $X \rightarrow A$ is not OK otherwise
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>413-555-1234</td>
<td>Amherst</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>413-555-6543</td>
<td>Amherst</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>

What is the key? \{SSN, PhoneNumber\}

Hence \( SSN \rightarrow \text{Name,City} \) is a “bad” dependency
Key or Keys?

Can we have more than one key?

Given R(A,B,C) define FD’s, s.t. there are two or more keys
Key or Keys?

Can we have more than one key?

Given R(A,B,C) define FD’s, s.t. there are two or more keys

AB \rightarrow C
BC \rightarrow A

or

A \rightarrow BC
B \rightarrow AC

what are the keys here?

Can you design FDs such that there are three keys?
Boyce-Codd Normal Form (BCNF)

A simple condition for removing anomalies from relations:

A relation R is in BCNF if:

If $A_1, ..., A_n \rightarrow B$ is a non-trivial dependency in R,
then $\{A_1, ..., A_n\}$ is a superkey for R

In other words: there are no “bad” FDs

Equivalently:
for all $X$, either ($X^+ = X$) or ($X^+ = \text{all attributes}$)
BCNF Decomposition Algorithm

repeat
   choose \( A_1, ..., A_m \rightarrow B_1, ..., B_n \) that violates BCNF
   split \( R \) into \( R_1(A_1, ..., A_m, B_1, ..., B_n) \) and \( R_2(A_1, ..., A_m, \text{[others]} \)
   continue with both \( R_1 \) and \( R_2 \)
until no more violations

Is there a 2-attribute relation that is not in BCNF?

In practice, we have a better algorithm (coming up)
Example (revisited)

<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>413-555-1234</td>
<td>Amherst</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>413-555-6543</td>
<td>Amherst</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>

What is the key? \{SSN, PhoneNumber\}

Hence \(SSN \rightarrow \text{Name,City}\) is a “bad” dependency
Example (revisited)

Let’s check anomalies:
  • Redundancy?
  • Update?
  • Delete?
Example Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

FD1: SSN → name, age
FD2: age → hairColor

Decompose into BCNF:
Example Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

FD1: SSN → name, age
FD2: age → hairColor

Decompose into BCNF:

What is the key? \{SSN, phoneNumber\}

But how to decompose?

Person(SSN, name, age)
Phone(SSN, hairColor, phoneNumber)

or

Person(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

SSN → name, age, hairColor

or ....
BCNF Decomposition Algorithm

BCNF_Decompose(R)

find X s.t.: X \neq X^+ \neq [all attributes]

**if** (not found) **then** “R is in BCNF”

**let** Y = X^+ - X
**let** Z = [all attributes] - X^+  
decompose R into R_1(X \cup Y) and R_2(X \cup Z)
continue to decompose recursively R_1 and R_2
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

FD1: SSN → name, age
FD2: age → hairColor

Find X s.t.: X ≠ X⁺ ≠ [all attributes]

Iteration 1: Person
SSN⁺ = SSN, name, age, hairColor
Decompose into: P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

Iteration 2: P
age⁺ = age, hairColor
Decompose: People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

What are the keys?
Example

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

\[ A \rightarrow B \]
\[ B \rightarrow C \]

\[ R_1(A,B,C) \]
\[ B^+ = BC \neq ABC \]

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

\[ R_2(A,D) \]

\[ R(A,B,C,D) \]
\[ A^+ = ABC \neq ABCD \]

What are the keys?

What happens if in R we first pick \( B^+ \)? Or \( AB^+ \)?
Decompositions in General

\[ R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \]
\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]
\[ R_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

R_1 = projection of R on \( A_1, \ldots, A_n, B_1, \ldots, B_m \)
R_2 = projection of R on \( A_1, \ldots, A_n, C_1, \ldots, C_p \)
Theory of Decomposition

Sometimes it is correct:

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Lossless decomposition
## Incorrect Decomposition

- Sometimes it is not:

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

### Lossy decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

What’s incorrect ??

Name → Price
Decompositions in General

If $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$
Then the decomposition is lossless

Note: don’t need $A_1, \ldots, A_n \rightarrow C_1, \ldots, C_p$

BCNF decomposition is always lossless. WHY?
General Decomposition Goals

1. Elimination of anomalies

2. Recoverability of information
   - Can we get the original relation back?

3. Preservation of dependencies
   - Want to enforce FDs without performing joins

Sometimes cannot decompose into BCNF without losing ability to check some FDs in single relation
BCNF and Dependencies

So, there is a BCNF violation, and we decompose.

In BCNF we lose the FD

Unit | Company | Product
--- | --- | ---

Unit → Company

Company, Product → Unit

Unit | Company
--- | ---

Unit → Company

No FDs

Unit | Product
--- | ---

Company, Product → Unit
3NF Motivation

A relation $R$ is in 3rd normal form if:
Whenever there is a nontrivial dep. $A_1, A_2, ..., A_n \rightarrow B$ for $R$, then $\{A_1, A_2, ..., A_n\}$ is a super-key for $R$, or $B$ is part of a key.

Tradeoffs:
- **BCNF**: no anomalies, but may lose some FDs
- **3NF**: keeps all FDs, but may have some anomalies