Fourth Normal Form

T. M. Murali

November 8, 2010
Attribute Independence in BCNF Schemas

Relation is Courses(Number, DeptName, Textbook, Professor).

Allow more than one textbook for the same course. Keep the same relation. Each professor uses every textbook in the course.

Is the relation in BCNF? Yes!

Is there any redundancy in the relation?
Attribute Independence in BCNF Schemas

Relation is Courses(Number, DeptName, Textbook, Professor).

- Relation is in BCNF.
Attribute Independence in BCNF Schemas

- Relation is Courses(Number, DeptName, Textbook, Professor).
- Allow more than professor to teach a course. Keep the same relation.
- Relation is Courses(Number, DeptName, Textbook, Professor).
- Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF?
Relation is Courses(Number, DeptName, Textbook, Professor).

Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF? No.
Relation is Courses(Number, DeptName, Textbook, Professor).

Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF? No.

Allow more than one textbook for the same course. Keep the same relation. Each professor uses every textbook in the course.
Attribute Independence in BCNF Schemas

Relation is Courses(Number, DeptName, Textbook, Professor).

Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF? No.

Allow more than one textbook for the same course. Keep the same relation. Each professor uses every textbook in the course. Is the relation in BCNF?
Attribute Independence in BCNF Schemas

Relation is Courses(Number, DeptName, Textbook, Professor).

- Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF? No.

- Allow more than one textbook for the same course. Keep the same relation. Each professor uses every textbook in the course. Is the relation in BCNF? Yes!
Attribute Independence in BCNF Schemas

- Relation is Courses(Number, DeptName, Textbook, Professor).
- Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF? No.
- Allow more than one textbook for the same course. Keep the same relation. Each professor uses every textbook in the course. Is the relation in BCNF? Yes!
- Is there any redundancy in the relation?
Attribute Independence in BCNF Schemas

- BCNF schemas can have redundancy, e.g., when we force two or more many-many relationships in a single relation.
- The relation is Courses(Number, DeptName, Textbook, Professor).
  - Each Course can have multiple required Textbooks.
  - Each Course can have multiple Professors.
  - A Professor uses every required textbook while teaching a Course.

The relation is in BCNF since there are no non-trivial FDs.

Is there any redundancy? Yes, in the Textbook and Professor attributes.
Attribute Independence in BCNF Schemas

- BCNF schemas can have redundancy, e.g., when we force two or more many-many relationships in a single relation.
- The relation is Courses(Number, DeptName, Textbook, Professor).
  - Each Course can have multiple required Textbooks.
  - Each Course can have multiple Professors.
  - A Professor uses every required textbook while teaching a Course.

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
</tbody>
</table>

- The relation is in BCNF since there are no non-trivial FDs.
Attribute Independence in BCNF Schemas

- BCNF schemas can have redundancy, e.g., when we force two or more many-many relationships in a single relation.
- The relation is Courses(Number, DeptName, Textbook, Professor).
  - Each Course can have multiple required Textbooks.
  - Each Course can have multiple Professors.
  - A Professor uses every required textbook while teaching a Course.
- The relation is in BCNF since there are no non-trivial FDs.
- Is there any redundancy?

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
</tbody>
</table>
BCNF schemas can have redundancy, e.g., when we force two or more many-many relationships in a single relation.

The relation is Courses(Number, DeptName, Textbook, Professor).

- Each Course can have multiple required Textbooks.
- Each Course can have multiple Professors.
- A Professor uses every required textbook while teaching a Course.

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
</tbody>
</table>

The relation is in BCNF since there are no non-trivial FDs.

Is there any redundancy? Yes, in the Textbook and Professor attributes.
We can remove the redundancy by decomposing Courses into Courses1(Number, DeptName, Textbook) and Courses2(Number, DeptName, Professor)
We can remove the redundancy by decomposing Courses into Courses1(Number, DeptName, Textbook) and Courses2(Number, DeptName, Professor).

FDs and BCNF are not rich enough to express these types of redundancies.
Multi-valued Dependencies

- A multi-valued dependency (MVD or MD) is an assertion that two sets of attributes are independent of each other.
- The *multi-valued dependency* \(A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m\) holds in a relation \(R\) if in every instance of \(R\), for every pair of tuples \(t\) and \(u\) in \(R\) that agree on all the \(A\)'s, we can find a tuple \(v\) in \(R\) that agrees
  1. with both \(t\) and \(u\) on the \(A_i\)s,
  2. with \(t\) on the \(B_j\)s, and
  3. with \(u\) on all those attributes of \(R\) that are not \(A_i\)s or \(B_j\)s.

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
</tbody>
</table>
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
</tr>
</tbody>
</table>
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
</tr>
</tbody>
</table>

> Number DeptName $\rightarrow$ Textbook is an MD.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td>u</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

- **Number DeptName → Textbook** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees
  1. with both \( t \) and \( u \) on Number and DeptName,
  2. with \( t \) on Textbook, and with \( u \) on Professor.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th>t</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td></td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Number DeptName → Textbook** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees
1. with both \( t \) and \( u \) on Number and DeptName,
2. with \( t \) on Textbook, and with \( u \) on Professor.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td>u</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

- **Number DeptName → Textbook** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees
  1. with both \( t \) and \( u \) on Number and DeptName,
  2. with \( t \) on Textbook, and with \( u \) on Professor.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th>T. M. Murali November 8, 2010 CS 4604: Fourth Normal Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td>v</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td>u</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

▶ **Number DeptName → Textbook** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees

1. with both $t$ and $u$ on Number and DeptName,
2. with $t$ on Textbook, and with $u$ on Professor.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
</tr>
</tbody>
</table>

- **Number DeptName → Textbook** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees
  1. with both \( t \) and \( u \) on Number and DeptName,
  2. with \( t \) on Textbook, and with \( u \) on Professor.

- **Number DeptName → Professor** is an MD.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
</tr>
</tbody>
</table>

- **Number DeptName → Textbook** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees
  1. with both $t$ and $u$ on Number and DeptName,
  2. with $t$ on Textbook, and with $u$ on Professor.

- **Number DeptName → Professor** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees
  1. with both $t$ and $u$ on Number and DeptName,
  2. with $t$ on Professor, and with $u$ on Textbook.
**Example of an MD**

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td><strong>FCDB</strong></td>
<td>Ullman</td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td>u</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td><strong>FCDB</strong></td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

- **Number DeptName → Textbook** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees
  1. with both \( t \) and \( u \) on Number and DeptName,
  2. with \( t \) on Textbook, and with \( u \) on Professor.

- **Number DeptName → Professor** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees
  1. with both \( t \) and \( u \) on Number and DeptName,
  2. with \( t \) on Professor, and with \( u \) on Textbook.
### Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

- **Number DeptName → Textbook** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees
  1. with both $t$ and $u$ on Number and DeptName,
  2. with $t$ on Textbook, and with $u$ on Professor.

- **Number DeptName → Professor** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees
  1. with both $t$ and $u$ on Number and DeptName,
  2. with $t$ on Professor, and with $u$ on Textbook.
### Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td>u</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

► **Number DeptName → Textbook** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees

1. with both \( t \) and \( u \) on Number and DeptName,
2. with \( t \) on Textbook, and with \( u \) on Professor.

► **Number DeptName → Professor** is an MD. For every pair of tuples \( t \) and \( u \) that agree on Number and DeptName, we can find a tuple \( v \) that agrees

1. with both \( t \) and \( u \) on Number and DeptName,
2. with \( t \) on Professor, and with \( u \) on Textbook.
Example of an MD

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>Textbook</th>
<th>Professor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Ullman</td>
<td>t</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Ullman</td>
<td>v</td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>FCDB</td>
<td>Widom</td>
<td></td>
</tr>
<tr>
<td>4604</td>
<td>CS</td>
<td>SQL Made Easy</td>
<td>Widom</td>
<td>u</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Ullman</td>
<td></td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>Widom</td>
<td></td>
</tr>
</tbody>
</table>

- **Number DeptName $\rightarrow$ Textbook** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees
  1. with both $t$ and $u$ on Number and DeptName,
  2. with $t$ on Textbook, and with $u$ on Professor.

- **Number DeptName $\rightarrow$ Professor** is an MD. For every pair of tuples $t$ and $u$ that agree on Number and DeptName, we can find a tuple $v$ that agrees
  1. with both $t$ and $u$ on Number and DeptName,
  2. with $t$ on Professor, and with $u$ on Textbook.
Fun Facts About MDs

- Given tuples \( t, u, \) and \( v \) that satisfy an MD, we can infer the existence of another tuple \( w \) that agrees
  1. with both \( t \) and \( u \) on \( A \)'s,
  2. with \( u \) on the \( B \)'s, and
  3. with \( t \) on all those attributes of \( R \) that are not \( A \)'s or \( B \)'s.
Rules for Manipulating MDs

- **FD promotion**: Every FD $A \rightarrow B$ is an MD $A \rightarrow B$.

- **Definition of keys depends on FDs and not on MDs.**

- **Trivial MVDs**:
  1. If $A \rightarrow B$ is an MD, then $A \rightarrow AB$ is also an MD.
  2. If $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_m$ make up all the attributes of a relation, then $A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$ holds in the relation.

- **Transitive rule**: if $A \rightarrow B$ and $B \rightarrow C$ are MDs, then $A \rightarrow C$ is an MD.

- **Complementation rule**: if $A \rightarrow B$, then $A \rightarrow C$ is an MD, where $C$ is the set of all attributes not in the MD.

- **The splitting rule does not hold**: If $A \rightarrow BC$ is an MD, then it is not true that $A \rightarrow B$ and $A \rightarrow C$ are MDs.
Rules for Manipulating MDs

- **FD promotion**: Every FD $A \rightarrow B$ is an MD $A \rightarrow B$. Proof: make $u$ and $v$ the same tuple.
Rules for Manipulating MDs

- *FD promotion*: Every FD $A \rightarrow B$ is an MD $A \rightarrow B$. Proof: make $u$ and $v$ the same tuple.
  - Definition of keys depends on FDs and not on MDs.
- *Trivial MVDs:*
  1. If $A \rightarrow B$ is an MD, then $A \rightarrow AB$ is also an MD.
  2. If $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_m$ make up all the attributes of a relation, then $A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$ holds in the relation.
Rules for Manipulating MDs

- **FD promotion:** Every FD $A \rightarrow B$ is an MD $A \rightarrow B$. Proof: make $u$ and $v$ the same tuple.
  - Definition of keys depends on FDs and not on MDs.

- **Trivial MVDs:**
  1. If $A \rightarrow B$ is an MD, then $A \rightarrow AB$ is also an MD.
  2. If $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_m$ make up all the attributes of a relation, then $A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$ holds in the relation.

- **Transitive rule:** if $A \rightarrow B$ and $B \rightarrow C$ are MDs, then $A \rightarrow C$ is an MD.
Rules for Manipulating MDs

- **FD promotion**: Every FD $A \rightarrow B$ is an MD $A \rightarrow B$. Proof: make $u$ and $v$ the same tuple.
  - Definition of keys depends on FDs and not on MDs.

- **Trivial MVDs**:
  1. If $A \rightarrow B$ is an MD, then $A \rightarrow AB$ is also an MD.
  2. If $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_m$ make up all the attributes of a relation, then $A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$ holds in the relation.

- **Transitive rule**: if $A \rightarrow B$ and $B \rightarrow C$ are MDs, then $A \rightarrow C$ is an MD.

- **Complementation rule**: if $A \rightarrow B$, then $A \rightarrow C$ is an MD, where $C$ is the set of all attributes not in the MD.
Rules for Manipulating MDs

- **FD promotion:** Every FD $A \rightarrow B$ is an MD $A \rightarrow B$. Proof: make $u$ and $v$ the same tuple.
  - Definition of keys depends on FDs and not on MDs.

- **Trivial MVDs:**
  1. If $A \rightarrow B$ is an MD, then $A \rightarrow AB$ is also an MD.
  2. If $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_m$ make up all the attributes of a relation, then $A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$ holds in the relation.

- **Transitive rule:** if $A \rightarrow B$ and $B \rightarrow C$ are MDs, then $A \rightarrow C$ is an MD.

- **Complementation rule:** if $A \rightarrow B$, then $A \rightarrow C$ is an MD, where $C$ is the set of all attributes not in the MD.

- The splitting rule does not hold! If $A \rightarrow BC$ is an MD, then it is not true that $A \rightarrow B$ and $A \rightarrow C$ are MDs.
A relation $R$ is in fourth normal form (4NF) if for every non-trivial MD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$, \{\footnotesize{$A_1$, $A_2$, \ldots, $A_n$}\} is a superkey.
Fourth Normal Form

- A relation $R$ is in fourth normal form (4NF) if for every non-trivial MD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$, \{A_1, A_2, \ldots, A_n\} is a superkey.
- A relation in 4NF is also in BCNF since an FD is a special case of an MD.
Decomposition into 4NF

- Suppose $R$ is a relation with a set $X$ of attributes and
- $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ violates 4NF.
- Decompose $R$ into two relations whose attributes are
  1. the $A$s and the $B$s, i.e., $\{A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m\}$ and
  2. all the attributes of $R$ that are not $B$'s, i.e., $X - \{B_1, B_2, \ldots, B_m\}$.
  3. Recursively check if the new relations are in 4NF, and decompose them if necessary.
Decomposition into 4NF

- Suppose $R$ is a relation with a set $X$ of attributes and $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ violates 4NF.
- Decompose $R$ into two relations whose attributes are
  1. the $A$s and the $B$s, i.e., $\{A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m\}$ and
  2. all the attributes of $R$ that are not $B$'s, i.e., $X - \{B_1, B_2, \ldots, B_m\}$.
  3. Recursively check if the new relations are in 4NF, and decompose them if necessary.
- Projecting MDs: need a method to discover new FDs.
- Date-Fagin theorem: if a relation schema is in BCNF and has a key with one attribute, then it is in 4NF.
Discovering New FDs and MDs

- Given a set of MDs and FDs, what new FDs and MDs follow?
- Algorithm is similar to the chase algorithm to determine if a decomposition is lossless join.
Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs.
- An alternative is to use the chase process.

\[
R (A, B, C, D) \text{ satisfies the FDs } AD \rightarrow C \text{ and } DC \rightarrow B.
\]

- Show that the FD \( AD \rightarrow B \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) and in \( D \) but are different in \( B \).
- Apply the FDs repeatedly to equate values of attributes.
- Stop either when both tuples agree in \( B \) (the FD holds) or no more FDs can be applied (the FD does not hold).

\[
\begin{array}{cccc}
A & B & C & D \\
\hline \\
a & b & 1 & d \\
a & b & 2 & d \\
\end{array}
\]

- Since \( AD \rightarrow C \), \( c_1 = c_2 \).
- Now since \( DC \rightarrow B \), \( b_1 = b_2 \), proving \( AD \rightarrow B \).
Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs.
- An alternative is to use the chase process.
- $R(A, B, C, D)$ satisfies the FDs $AD \rightarrow C$ and $DC \rightarrow B$.
- Show that the FD $AD \rightarrow B$ holds in $R$. 

$\begin{array}{cccc}
A & B & C & D \\
\text{a} & \text{b} & \text{c} & \text{d} \\
\text{a} & \text{b} & \text{c} & \text{d} \\
\end{array}$

- Since $AD \rightarrow C$, $c_1 = c_2$.
- Now since $DC \rightarrow B$, $b_1 = b_2$, proving $AD \rightarrow B$. 

T. M. Murali November 8, 2010 CS 4604: Fourth Normal Form
Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs.
- An alternative is to use the chase process.
- \( R(A, B, C, D) \) satisfies the FDs \( AD \rightarrow C \) and \( DC \rightarrow B \).
- Show that the FD \( AD \rightarrow B \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) and in \( D \) but are different in \( B \).
- Apply the FDs repeatedly to equate values of attributes.
- Stop either when both tuples agree in \( B \) (the FD holds) or no more FDs can be applied (the FD does not hold).
Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs.
- An alternative is to use the chase process.
- \( R(A, B, C, D) \) satisfies the FDs \( AD \rightarrow C \) and \( DC \rightarrow B \).
- Show that the FD \( AD \rightarrow B \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) and in \( D \) but are different in \( B \).
- Apply the FDs repeatedly to equate values of attributes.
- Stop either when both tuples agree in \( B \) (the FD holds) or no more FDs can be applied (the FD does not hold).

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b_1 & c_1 & d \\
a & b_2 & c_2 & d \\
\end{array}
\]

Since \( AD \rightarrow C \), \( c_1 = c_2 \).

Now since \( DC \rightarrow B \), \( b_1 = b_2 \), proving \( AD \rightarrow B \).
Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs.
- An alternative is to use the chase process.
- \( R(A, B, C, D) \) satisfies the FDs \( AD \rightarrow C \) and \( DC \rightarrow B \).
- Show that the FD \( AD \rightarrow B \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) and in \( D \) but are different in \( B \).
- Apply the FDs repeatedly to equate values of attributes.
- Stop either when both tuples agree in \( B \) (the FD holds) or no more FDs can be applied (the FD does not hold).

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b_1 )</td>
<td>( c_1 )</td>
<td>( d )</td>
</tr>
<tr>
<td>( a )</td>
<td>( b_2 )</td>
<td>( c_2 )</td>
<td>( d )</td>
</tr>
</tbody>
</table>

- Since \( AD \rightarrow C \), \( c_1 = c_2 \).
Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs.
- An alternative is to use the chase process.
- \( R(A, B, C, D) \) satisfies the FDs \( AD \rightarrow C \) and \( DC \rightarrow B \).
- Show that the FD \( AD \rightarrow B \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) and in \( D \) but are different in \( B \).
- Apply the FDs repeatedly to equate values of attributes.
- Stop either when both tuples agree in \( B \) (the FD holds) or no more FDs can be applied (the FD does not hold).

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b_1 & c_1 & d \\
a & b_2 & c_2 & d \\
\end{array}
\]

- Since \( AD \rightarrow C \), \( c_1 = c_2 \).
- Now since \( DC \rightarrow B \), \( b_1 = b_2 \), proving \( AD \rightarrow B \).
We can use the chase process to infer new FDs when given FDs and MDs as input.

\[
\begin{array}{cccc}
A & B & C & D \\
a & b & 1 & c \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
a & b & 2 & c \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
t & a & b & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
t & a & b & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
v & a & b & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
v & a & b & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
w & a & b & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
w & a & b & 2 \\
\end{array}
\]

Since \(A \rightarrow BC\), \(R\) must also contain the tuples \(v = (a, b_1, c_1, d_2)\) and \(w = (a, b_2, c_2, d_1)\).

Applying \(D \rightarrow C\) to \(t\) and \(w\), we see that \(c_1 = c_2\), proving the FD.
We can use the chase process to infer new FDs when given FDs and MDs as input.

$R(A, B, C, D)$ satisfies the FD $D \rightarrow C$ and the MD $A \rightarrow BC$.

Show that the FD $A \rightarrow C$ holds in $R$.

Since $A \rightarrow BC$, $R$ must also contain the tuples $v = (a, b_1, c_1, d_2)$ and $w = (a, b_2, c_2, d_1)$.

Applying $D \rightarrow C$ to $t$ and $w$, we see that $c_1 = c_2$, proving the FD.
We can use the chase process to infer new FDs when given FDs and MDs as input.

- $R(A, B, C, D)$ satisfies the FD $D \rightarrow C$ and the MD $A \rightarrow BC$.
- Show that the FD $A \rightarrow C$ holds in $R$.
- Start with a tableau containing two tuples that agree in $A$ but are different in $C$.
- Apply the FDs to equate values of attributes.
- Apply the MDs to infer new tuples in the relation.
- Stop either when both tuples agree in $C$ (the FD holds) or no more FDs/MDs can be applied (the FD does not hold).
Chase for Discovering New FDs (2)

- We can use the chase process to infer new FDs when given FDs and MDs as input.
- \( R(A, B, C, D) \) satisfies the FD \( D \rightarrow C \) and the MD \( A \rightarrow BC \).
- Show that the FD \( A \rightarrow C \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) but are different in \( C \).
- Apply the FDs to equate values of attributes.
- Apply the MDs to infer new tuples in the relation.
- Stop either when both tuples agree in \( C \) (the FD holds) or no more FDs/MDs can be applied (the FD does not hold).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b_1</td>
<td>c_1</td>
<td>d_1</td>
</tr>
<tr>
<td>a</td>
<td>b_2</td>
<td>c_2</td>
<td>d_2</td>
</tr>
</tbody>
</table>
Chase for Discovering New FDs (2)

- We can use the chase process to infer new FDs when given FDs and MDs as input.
- \( R(A, B, C, D) \) satisfies the FD \( D \rightarrow C \) and the MD \( A \rightarrow BC \).
- Show that the FD \( A \rightarrow C \) holds in \( R \).
- Start with a tableau containing two tuples that agree in \( A \) but are different in \( C \).
- Apply the FDs to equate values of attributes.
- Apply the MDs to infer new tuples in the relation.
- Stop either when both tuples agree in \( C \) (the FD holds) or no more FDs/MDs can be applied (the FD does not hold).

- Since \( A \rightarrow BC \), \( R \) must also contain the tuples

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b₁</td>
<td>c₁</td>
<td>d₁</td>
</tr>
<tr>
<td>a</td>
<td>b₂</td>
<td>c₂</td>
<td>d₂</td>
</tr>
</tbody>
</table>
We can use the chase process to infer new FDs when given FDs and MDs as input.

\[ R(A, B, C, D) \] satisfies the FD \( D \rightarrow C \) and the MD \( A \rightarrow BC \).

Show that the FD \( A \rightarrow C \) holds in \( R \).

Start with a tableau containing two tuples that agree in \( A \) but are different in \( C \).

Apply the FDs to equate values of attributes.

Apply the MDs to infer new tuples in the relation.

Stop either when both tuples agree in \( C \) (the FD holds) or no more FDs/MDs can be applied (the FD does not hold).

Since \( A \rightarrow BC \), \( R \) must also contain the tuples \( v = (a, b_1, c_1, d_2) \) and \( w = (a, b_2, c_2, d_1) \).

Applying \( D \rightarrow C \) to \( t \) and \( w \), we see that \( c_1 = c_2 \), proving the FD.
Chase for Discovering New MDs

- We can use the same process to new discover new MDs as well!
- Relation $R(A, B, C, D)$ satisfies the FD $A \rightarrow B$ and MD $B \rightarrow C$. Which new MDs hold in $R$?

Let us try to prove that $A \rightarrow C$ holds.

Create a tableau with two different tuples that agree in $A$ and differ in $C$. If $A \rightarrow C$ holds, it implies the existence of two other tuples in $R$.

Use the FDs and MDs to prove the existence of these tuples.

To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is $(a, b, c, d)$.

- Since $A \rightarrow B$, $b = b_1$.
- Since $B \rightarrow C$, $R$ also contains tuples $(a, b, c_2, d_1)$ and $(a, b, c, d)$, proving $A \rightarrow C$.

Projecting MDs into a new relation: read Chapter 3.7.4 of the textbook.
We can use the same process to new discover new MDs as well!

Relation $R(A, B, C, D)$ satisfies the FD $A \rightarrow B$ and MD $B \rightarrow C$.
Which new MDs hold in $R$?

Let us try to prove that $A \rightarrow C$ holds.

Create a tableau with two different tuples that agree in $A$ and differ in $C$. If $A \rightarrow C$ holds, it implies the existence of two other tuples in $R$.

Use the FDs and MDs to prove the existence of these tuples.

To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is $(a, b, c, d)$. Since $A \rightarrow B$, $b = b_1$. Since $B \rightarrow C$, $R$ also contains tuples $(a, b, c_2, d_1)$ and $(a, b, c, d)$, proving $A \rightarrow C$.

Projecting MDs into a new relation: read Chapter 3.7.4 of the textbook.
Chase for Discovering New MDs

- We can use the same process to new discover new MDs as well!
- Relation $R(A, B, C, D)$ satisfies the FD $A \rightarrow B$ and MD $B \twoheadrightarrow C$. Which new MDs hold in $R$?
- Let us try to prove that $A \twoheadrightarrow C$ holds.
- Create a tableau with two different tuples that agree in $A$ and differ in $C$. If $A \twoheadrightarrow C$ holds, it implies the existence of two other tuples in $R$.
- Use the FDs and MDs to prove the existence of these tuples.
- To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is $(a, b, c, d)$. 

Projecting MDs into a new relation: read Chapter 3.7.4 of the textbook.
Chase for Discovering New MDs

- We can use the same process to new discover new MDs as well!
- Relation $R(A, B, C, D)$ satisfies the FD $A \rightarrow B$ and MD $B \rightarrow C$. Which new MDs hold in $R$?
- Let us try to prove that $A \rightarrow C$ holds.
- Create a tableau with two different tuples that agree in $A$ and differ in $C$. If $A \rightarrow C$ holds, it implies the existence of two other tuples in $R$.
- Use the FDs and MDs to prove the existence of these tuples.
- To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is $(a, b, c, d)$.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
<td>$D$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b_1$</td>
<td>$c$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c_2$</td>
<td>$d$</td>
</tr>
</tbody>
</table>

- Since $A \rightarrow B$, $b = b_1$. 

T. M. Murali

November 8, 2010

CS 4604: Fourth Normal Form
Chase for Discovering New MDs

- We can use the same process to new discover new MDs as well!
- Relation $R(A, B, C, D)$ satisfies the FD $A \rightarrow B$ and MD $B \rightarrow C$. Which new MDs hold in $R$?
- Let us try to prove that $A \rightarrow C$ holds.
- Create a tableau with two different tuples that agree in $A$ and differ in $C$. If $A \rightarrow C$ holds, it implies the existence of two other tuples in $R$.
- Use the FDs and MDs to prove the existence of these tuples.
- To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is $(a, b, c, d)$.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$b_1$</td>
<td>c</td>
<td>$d_1$</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>$c_2$</td>
<td>d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>t</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>u</td>
</tr>
</tbody>
</table>

- Since $A \rightarrow B$, $b = b_1$.
- Since $B \rightarrow C$, $R$ also contains tuples $(a, b, c_2, d_1)$ and $(a, b, c, d)$, proving $A \rightarrow C$. 
Chase for Discovering New MDs

▶ We can use the same process to new discover new MDs as well!
▶ Relation $R(A, B, C, D)$ satisfies the FD $A \rightarrow B$ and MD $B \rightarrow C$. Which new MDs hold in $R$?
▶ Let us try to prove that $A \rightarrow C$ holds.
▶ Create a tableau with two different tuples that agree in $A$ and differ in $C$. If $A \rightarrow C$ holds, it implies the existence of two other tuples in $R$.
▶ Use the FDs and MDs to prove the existence of these tuples.
▶ To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is $(a, b, c, d)$.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b_1$</td>
<td>$c$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$c_2$</td>
<td>$d$</td>
</tr>
</tbody>
</table>

▶ Since $A \rightarrow B$, $b = b_1$.
▶ Since $B \rightarrow C$, $R$ also contains tuples $(a, b, c_2, d_1)$ and $(a, b, c, d)$, proving $A \rightarrow C$. 
Chase for Discovering New MDs

- We can use the same process to new discover new MDs as well!
- Relation \( R(A, B, C, D) \) satisfies the FD \( A \rightarrow B \) and MD \( B \rightarrow C \). Which new MDs hold in \( R \)?
- Let us try to prove that \( A \rightarrow C \) holds.
- Create a tableau with two different tuples that agree in \( A \) and differ in \( C \). If \( A \rightarrow C \) holds, it implies the existence of two other tuples in \( R \).
- Use the FDs and MDs to prove the existence of these tuples.
- To simplify, name the attributes in the initial two tuples such that one of the new tuples we will prove is \((a, b, c, d)\).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>b₁</td>
<td>c</td>
<td>d₁</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>b</td>
<td>c₂</td>
<td>d</td>
</tr>
</tbody>
</table>

- Since \( A \rightarrow B \), \( b = b₁ \).
- Since \( B \rightarrow C \), \( R \) also contains tuples \((a, b, c₂, d₁)\) and \((a, b, c, d)\), proving \( A \rightarrow C \).
- Projecting MDs into a new relation: read Chapter 3.7.4 of the textbook.
4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF.

BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF.

<table>
<thead>
<tr>
<th>Property</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates redundancy due to FDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eliminates redundancy due to MDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preserves FDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preserves MDs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF.

BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF.

<table>
<thead>
<tr>
<th>Property</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates redundancy due to FDs</td>
<td>Maybe</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates redundancy due to MDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preserves FDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preserves MDs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Relationships Among Normal Forms

- 4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF.
- BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF.

<table>
<thead>
<tr>
<th>Property</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates redundancy due to FDs</td>
<td>Maybe</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates redundancy due to MDs</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Preserves FDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preserves MDs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF.

BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF.

<table>
<thead>
<tr>
<th>Property</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates redundancy due to FDs</td>
<td>Maybe</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates redundancy due to MDs</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Preserves FDs</td>
<td>Yes</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Preserves MDs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF.

BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF.

<table>
<thead>
<tr>
<th>Property</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates redundancy due to FDs</td>
<td>Maybe</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates redundancy due to MDs</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Preserves FDs</td>
<td>Yes</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Preserves MDs</td>
<td>Maybe</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
$k$NFs

- First Normal Form: each attribute is atomic.
$k$NFs

- First Normal Form: each attribute is atomic.
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key.
$k$NFs

- First Normal Form: each attribute is atomic.
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key.
- Third Normal Form: we just discussed it.
- Fourth Normal Form: we just discussed it.
- Fifth Normal Form: outside the scope of CS 4604.
- Sixth Normal Form: different versions exist. One version is newly developed for temporal databases.
- Seventh Normal Form: your ticket to fame and fortune.
$k$NFs

- First Normal Form: each attribute is atomic.
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key.
- Third Normal Form: we just discussed it.
- Fourth Normal Form: we just discussed it.
$k$NFs

- First Normal Form: each attribute is atomic.
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key.
- Third Normal Form: we just discussed it.
- Fourth Normal Form: we just discussed it.
- Fifth Normal Form: outside the scope of CS 4604.
- Sixth Normal Form: different versions exist. One version is newly developed for temporal databases.
- Seventh Normal Form: your ticket to fame and fortune.
**kNFs**

- First Normal Form: each attribute is atomic.
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key.
- Third Normal Form: we just discussed it.
- Fourth Normal Form: we just discussed it.
- Fifth Normal Form: outside the scope of CS 4604.
- Sixth Normal Form: different versions exist. One version is newly developed for temporal databases.
$k$NFS

- First Normal Form: each attribute is atomic.
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key.
- Third Normal Form: we just discussed it.
- Fourth Normal Form: we just discussed it.
- Fifth Normal Form: outside the scope of CS 4604.
- Sixth Normal Form: different versions exist. One version is newly developed for temporal databases.
- Seventh Normal Form: your ticket to fame and fortune.