Functional Dependencies

T. M. Murali

October 18, 25, 2010
Midterm Review

The database server setup is driving me nuts!

The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.

Homework grading is strict/uneven.

Increase amount of examples/hands-on work.

Make the lectures more interactive.

Office hours change a lot.

Wish the project was free form.

It would be nice to have a table of RA symbols and E/R diagram shapes.
Midterm Review

- The database server set up is driving me NUTS!
Midterm Review

- The database server set up is driving me NUTS!
- The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.
Midterm Review

- The database server set up is driving me NUTS!
- The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.
- Homework grading is strict/uneven.
Midterm Review

- The database server set up is driving me NUTS!
- The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.
- Homework grading is strict/uneven.
- Increase amount of examples/hands-on work.
The database server setup is driving me nuts!

The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.

Homework grading is strict/uneven.

Increase amount of examples/hands-on work.

Make the lectures more interactive.
Midterm Review

▶ The database server set up is driving me NUTS!
▶ The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.
▶ Homework grading is strict/uneven.
▶ Increase amount of examples/hands-on work.
▶ Make the lectures more interactive.
▶ Office hours change a lot.
Midterm Review

- The database server set up is driving me NUTS!
- The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.
- Homework grading is strict/uneven.
- Increase amount of examples/hands-on work.
- Make the lectures more interactive.
- Office hours change a lot.
- Wish the project was free form.
Midterm Review

- The database server set up is driving me NUTS!
- The class is slow (no 3NF yet). Why are we doing design before learning normalisation? It seems odd to teach relational models after SQL.
- Homework grading is strict/uneven.
- Increase amount of examples/hands-on work.
- Make the lectures more interactive.
- Office hours change a lot.
- Wish the project was free form.
- It would be nice to have a table of RA symbols and E/R diagram shapes.
Example

Convert to relations:

- Students(Id, Name)
- Advisors(Id, Name)
- Advises(StudentId, AdvisorId)
- Favourite(StudentId, AdvisorId)
Example

Convert to relations:

Students(Id, Name)
Advisors(Id, Name)
Advises(StudentId, AdvisorId)
Favourite(StudentId, AdvisorId)
Example

- Convert to relations:
  - Students(Id, Name)
  - Advisors(Id, Name)
  - Advises(StudentId, AdvisorId)
  - Favourite(StudentId, AdvisorId)
- Suppose we perversely decide to convert Students, Advises, and Favourite into one relation.
  - Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes?
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student's Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  \[
  \text{Id} \rightarrow \text{Name} \\
  \text{Id} \rightarrow \text{FavouriteAdvisorId}
  \]
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

▶ If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.

Id → Name
Id → FavouriteAdvisorId
AdvisorId →
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  \[ Id \rightarrow Name \]

  \[ Id \rightarrow FavouriteAdvisorId \]

  \[ AdvisorId \rightarrow AdvisorName \]
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  \[ \text{Id} \rightarrow \text{Name} \]
  
  \[ \text{Id} \rightarrow \text{FavouriteAdvisorId} \]
  
  \[ \text{AdvisorId} \rightarrow \text{AdvisorName} \]

- Can we say \( \text{Id} \rightarrow \text{AdvisorId} \)?
**Example of a Bad Relation**

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  \[ \text{Id} \rightarrow \text{Name} \]
  
  \[ \text{Id} \rightarrow \text{FavouriteAdvisorId} \]
  
  \[ \text{AdvisorId} \rightarrow \text{AdvisorName} \]

- Can we say \( \text{Id} \rightarrow \text{AdvisorId} \)? NO! Id is not a key for the Students relation.
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  $Id \rightarrow Name$
  
  $Id \rightarrow FavouriteAdvisorId$
  
  $AdvisorId \rightarrow AdvisorName$
  
- Can we say $Id \rightarrow AdvisorId$? NO! Id is not a key for the Students relation.
  
- What is the key for the Students?
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student’s Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  \[
  \text{Id} \rightarrow \text{Name} \\
  \text{Id} \rightarrow \text{FavouriteAdvisorId} \\
  \text{AdvisorId} \rightarrow \text{AdvisorName}
  \]

- Can we say \text{Id} \rightarrow \text{AdvisorId}? NO! Id is not a key for the Students relation.

- What is the key for the Students? \{Id, AdvisorId\}. 

T. M. Murali October 18, 25, 2010 CS 4604: Functional Dependencies
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- If you know a student's Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.
  
  \[ \text{Id} \rightarrow \text{Name} \]
  
  \[ \text{Id} \rightarrow \text{FavouriteAdvisorId} \]
  
  \[ \text{AdvisorId} \rightarrow \text{AdvisorName} \]

- Can we say \( \text{Id} \rightarrow \text{AdvisorId} \)? NO! Id is not a key for the Students relation.

- What is the key for the Students? \( \{ \text{Id}, \text{AdvisorId} \} \).

- Why is this relation “bad?”
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

▶ If you know a student's Id, can you determine the values of any other attributes? Name and FavouriteAdvisorId.

\[ \text{Id} \rightarrow \text{Name} \]
\[ \text{Id} \rightarrow \text{FavouriteAdvisorId} \]
\[ \text{AdvisorId} \rightarrow \text{AdvisorName} \]

▶ Can we say \( \text{Id} \rightarrow \text{AdvisorId} \)? NO! Id is not a key for the Students relation.

▶ What is the key for the Students? \( \{\text{Id, AdvisorId}\} \).

▶ Why is this relation “bad?” Parts of the key determine other attributes.
Motivation for Functional Dependencies

- Reason about constraints on attributes in a relation.
- Procedurally determine the keys of a relation.
- Detect when a relation has redundant information.
- Improve database designs systematically using normalisation.
Definition of Functional Dependency

- If $t$ is a tuple in a relation $R$ and $A$ is an attribute of $R$, then $t_A$ is the value of attribute $A$ in tuple $t$. 
Definition of Functional Dependency

- If $t$ is a tuple in a relation $R$ and $A$ is an attribute of $R$, then $t_A$ is the value of attribute $A$ in tuple $t$.
- The FD $\text{AdvisorId} \rightarrow \text{AdvisorName}$ holds in $R$ if in every instance of $R$, for every pair of tuples $t$ and $u$...
Definition of Functional Dependency

- If $t$ is a tuple in a relation $R$ and $A$ is an attribute of $R$, then $t_A$ is the value of attribute $A$ in tuple $t$.
- The FD $AdvisorId \rightarrow AdvisorName$ holds in $R$ if in every instance of $R$, for every pair of tuples $t$ and $u$
  if $t_{AdvisorId} = u_{AdvisorId}$, then $t_{AdvisorName} = u_{AdvisorName}$.
**Definition of Functional Dependencies**

A *functional dependency* (FD) on a relation $R$ is a statement

- If two tuples in $R$ agree on attributes $A_1, A_2, \ldots A_n$ then they agree on attribute $B$.
- Notation: $A_1 A_2 \ldots A_n \rightarrow B$
A functional dependency (FD) on a relation $R$ is a statement

- If two tuples in $R$ agree on attributes $A_1, A_2, \ldots, A_n$ then they agree on attribute $B$.
- Notation: $A_1 A_2 \ldots A_n \rightarrow B$
- FD says that for every pair of tuples $t$ and $u$ in any instance of $R$,
A functional dependency (FD) on a relation $R$ is a statement

- If two tuples in $R$ agree on attributes $A_1, A_2, \ldots, A_n$ then they agree on attribute $B$.
- Notation: $A_1A_2\ldots A_n \rightarrow B$
- FD says that for every pair of tuples $t$ and $u$ in any instance of $R$, if $t_{A_1} = u_{A_1}$ and $t_{A_2} = u_{A_2}$ and $\ldots t_{A_n} = u_{A_n}$, then $t_B = u_B$.
- The set of attributes $A_1, A_2, \ldots, A_n$ functionally determine $B$.
- An FD is a constraint on a single relation schema. It must hold on every instance of the relation.
- You cannot deduce an FD from a relation instance.
Examples of FDs

What FDs can we assert for the relation

Courses(Number, DeptName, CourseName, Classroom, Enrollment)

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>Databases</td>
<td>TORG 1020</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>Dance</td>
<td>Tree Dancing</td>
<td>Drillfield</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>English</td>
<td>The Basis of Data</td>
<td>Williams 44</td>
<td>45</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>MCB 114</td>
<td>100</td>
</tr>
<tr>
<td>2604</td>
<td>Physics</td>
<td>Dark Matter</td>
<td>Williams 44</td>
<td>100</td>
</tr>
</tbody>
</table>

Is Number → Enrollment an FD?
Examples of FDs

What FDs can we assert for the relation

Courses(Number, DeptName, CourseName, Classroom, Enrollment)

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>Databases</td>
<td>TORG 1020</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>Dance</td>
<td>Tree Dancing</td>
<td>Drillfield</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>English</td>
<td>The Basis of Data</td>
<td>Williams 44</td>
<td>45</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>MCB 114</td>
<td>100</td>
</tr>
<tr>
<td>2604</td>
<td>Physics</td>
<td>Dark Matter</td>
<td>Williams 44</td>
<td>100</td>
</tr>
</tbody>
</table>

Number DeptName → CourseName
Number DeptName → Classroom
Number DeptName → Enrollment
Examples of FDs

What FDs can we assert for the relation Courses(Number, DeptName, CourseName, Classroom, Enrollment)

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>Databases</td>
<td>TORG 1020</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>Dance</td>
<td>Tree Dancing</td>
<td>Drillfield</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>English</td>
<td>The Basis of Data</td>
<td>Williams 44</td>
<td>45</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>MCB 114</td>
<td>100</td>
</tr>
<tr>
<td>2604</td>
<td>Physics</td>
<td>Dark Matter</td>
<td>Williams 44</td>
<td>100</td>
</tr>
</tbody>
</table>

Number DeptName → CourseName
Number DeptName → Classroom
Number DeptName → Enrollment
Number DeptName → CourseName Classroom Enrollment
Examples of FDs

What FDs can we assert for the relation

Courses(Number, DeptName, CourseName, Classroom, Enrollment)

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>Databases</td>
<td>TORG 1020</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>Dance</td>
<td>Tree Dancing</td>
<td>Drillfield</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>English</td>
<td>The Basis of Data</td>
<td>Williams 44</td>
<td>45</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>MCB 114</td>
<td>100</td>
</tr>
<tr>
<td>2604</td>
<td>Physics</td>
<td>Dark Matter</td>
<td>Williams 44</td>
<td>100</td>
</tr>
</tbody>
</table>

Number DeptName → CourseName
Number DeptName → Classroom
Number DeptName → Enrollment
Number DeptName → CourseName Classroom Enrollment

Is Number → Enrollment an FD?
Where do FDs come from?

▶ “Keyness” of attributes.
▶ Domain and application constraints.
▶ Real world constraints, e.g.,

ProfessorID → Classroom
Where do FDs come from?

- “Keyness” of attributes.
- Domain and application constraints.
- Real world constraints, e.g., ProfessorID Time → Classroom
Definition of Keys

- FDs allow us to formally define keys.
- A set of attributes \( \{A_1, A_2, \ldots A_n\} \) is a key for a relation \( R \) if
  - **Uniqueness** \( \{A_1, A_2, \ldots A_n\} \) functionally determine all the other attributes of \( R \) and
  - **Minimality** no proper subset of \( \{A_1, A_2, \ldots A_n\} \) functionally determines all the other attributes of \( R \).
Definition of Keys

- FDs allow us to formally define keys.
- A set of attributes \( \{A_1, A_2, \ldots A_n\} \) is a key for a relation \( R \) if
  - **Uniqueness** \( \{A_1, A_2, \ldots A_n\} \) functionally determine all the other attributes of \( R \) and
  - **Minimality** no proper subset of \( \{A_1, A_2, \ldots A_n\} \) functionally determines all the other attributes of \( R \).
- A **superkey** is a set of attributes that has the uniqueness property but is not necessarily minimal.
- If a relation has multiple keys, specify one to be the **primary key**.
- Convention: in a relational schema, underline the attributes of the primary key.
- If a key has only one attribute \( A \), we say that \( A \) rather than \( \{A\} \) is a key.
Example of Keys

What is the key for Courses(Number, DeptName, CourseName, Classroom, Enrollment)?

The key is \{Number, DeptName\}. Why?

No proper subset of \{Number, DeptName\} has this property.
Example of Keys

What is the key for Courses(Number, DeptName, CourseName, Classroom, Enrollment)?

The key is {Number, DeptName}.
  - These attributes functionally determine every other attribute.
  - No proper subset of {Number, DeptName} has this property.
Example of Keys

What is the key for

Courses(\text{Number, DeptName, CourseName, Classroom, Enrollment})?

The key is \{Number, DeptName\}.

- These attributes functionally determine every other attribute.
- No proper subset of \{Number, DeptName\} has this property.

What is the key for

Teach(\text{Number, DepartmentName, ProfessorName, Classroom})?
Example of Keys

▶ What is the key for Courses(Number, DeptName, CourseName, Classroom, Enrollment)?

▶ The key is \{Number, DeptName\}.
  ▶ These attributes functionally determine every other attribute.
  ▶ No proper subset of \{Number, DeptName\} has this property.

▶ What is the key for Teach(Number, DepartmentName, ProfessorName, Classroom)?

▶ The key is \{Number, DepartmentName\}. Why?
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$: 
Keys in the Conversion from E/R to Relational Designs

► If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.

► If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  ► $R$ is many-many:
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$. 

Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$.
  - $R$ is many-one from $E$ to $F$:
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$.
  - $R$ is many-one from $E$ to $F$: key attributes of the relation are the key attributes of $E$. 
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$.
  - $R$ is many-one from $E$ to $F$: key attributes of the relation are the key attributes of $E$.
  - $R$ is one-one:
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$.
  - $R$ is many-one from $E$ to $F$: key attributes of the relation are the key attributes of $E$.
  - $R$ is one-one: key attributes of the relation are the key attributes of $E$ or of $F$. 

▶ If the relationship $R$ is multiway, we need to reason about the FDs that $R$ satisfies.
▶ There is no simple rule.
▶ If $R$ has an arrow towards entity set $E$, at least one key for the relation for $R$ excludes the key for $E$. 

T. M. Murali October 18, 25, 2010 CS 4604: Functional Dependencies
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$.
  - $R$ is many-one from $E$ to $F$: key attributes of the relation are the key attributes of $E$.
  - $R$ is one-one: key attributes of the relation are the key attributes of $E$ or of $F$.
- If the relationship $R$ is multiway
Keys in the Conversion from E/R to Relational Designs

- If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set.
- If the relation comes from a binary relationship $R$ between entity sets $E$ and $F$:
  - $R$ is many-many: key attributes of the relation are the key attributes of $E$ and of $F$.
  - $R$ is many-one from $E$ to $F$: key attributes of the relation are the key attributes of $E$.
  - $R$ is one-one: key attributes of the relation are the key attributes of $E$ or of $F$.
- If the relationship $R$ is multiway, we need to reason about the FDs that $R$ satisfies.
  - There is no simple rule.
  - If $R$ has an arrow towards entity set $E$, at least one key for the relation for $R$ excludes the key for $E$. 
Rules for Manipulating FDs

- Learn how to reason about FDs.
- Define rules for deriving new FDs from a given set of FDs.
- Next class: use these rules to remove “anomalies” from relational designs.
Rules for Manipulating FDs

- Learn how to reason about FDs.
- Define rules for deriving new FDs from a given set of FDs.
- Next class: use these rules to remove “anomalies” from relational designs.
- Example: A relation $R$ with attributes $A$, $B$, and $C$, satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. What other FDs does it satisfy?
Rules for Manipulating FDs

- Learn how to reason about FDs.
- Define rules for deriving new FDs from a given set of FDs.
- Next class: use these rules to remove “anomalies” from relational designs.
- Example: A relation $R$ with attributes $A$, $B$, and $C$, satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. What other FDs does it satisfy? $A \rightarrow C$. 
Learn how to reason about FDs.
Define rules for deriving new FDs from a given set of FDs.
Next class: use these rules to remove “anomalies” from relational designs.
Example: A relation $R$ with attributes $A$, $B$, and $C$, satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. What other FDs does it satisfy? $A \rightarrow C$.
What is the key for $R$?
Rules for Manipulating FDs

- Learn how to reason about FDs.
- Define rules for deriving new FDs from a given set of FDs.
- Next class: use these rules to remove “anomalies” from relational designs.
- Example: A relation $R$ with attributes $A$, $B$, and $C$, satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. What other FDs does it satisfy? $A \rightarrow C$.
- What is the key for $R$? $A$, because $A \rightarrow B$ and $A \rightarrow C$. 
Equivalence of FDs

- An FD $F$ follows from a set of FDs $T$ if every relation instance that satisfies all the FDs in $T$ also satisfies $F$.
  - $A \rightarrow C$ follows from $T = \{ A \rightarrow B, B \rightarrow C \}$.
  - Does $T$ follow from $S$?
Equivalence of FDs

▶ An FD $F$ follows from a set of FDs $T$ if every relation instance that satisfies all the FDs in $T$ also satisfies $F$.
  ▶ $A \rightarrow C$ follows from $T = \{A \rightarrow B, B \rightarrow C\}$.
  ▶ Does $T$ follow from $S$?

▶ Two sets of FDs $S$ and $T$ are equivalent if each FD in $S$ follows from $T$ and each FD in $T$ follows from $S$.
  ▶ $S = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ and $T = \{A \rightarrow B, B \rightarrow C\}$ are equivalent.
Equivalence of FDs

- An FD $F$ follows from a set of FDs $T$ if every relation instance that satisfies all the FDs in $T$ also satisfies $F$.
  - $A \rightarrow C$ follows from $T = \{A \rightarrow B, B \rightarrow C\}$.
  - Does $T$ follow from $S$?

- Two sets of FDs $S$ and $T$ are equivalent if each FD in $S$ follows from $T$ and each FD in $T$ follows from $S$.
  - $S = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ and $T = \{A \rightarrow B, B \rightarrow C\}$ are equivalent.

- These notions are useful in deriving new FDs from a given set of FDs.
Splitting and Combining FDs

- The set of FDs
  \[ A_1 A_2 \ldots A_n \rightarrow B_1 \]
  \[ A_1 A_2 \ldots A_n \rightarrow B_2 \]
  \[ \vdots \]
  \[ A_1 A_2 \ldots A_n \rightarrow B_m \]

  is equivalent to the FD
  \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m. \]

- This equivalence implies two rules.
  - *Splitting rule*
  - *Combining rule*
  - These rules work because all the FDs in $S$ and $T$ have identical left hand sides.
Splitting and Combining FDs

- Can we split and combine left hand sides of FDs?
Splitting and Combining FDs

- Can we split and combine left hand sides of FDs?
- For the relation Courses, is the FD
  \[ \text{Number } \text{DeptName} \rightarrow \text{CourseName} \]
equivalent to the set of FDs
  \[ \{ \text{Number } \rightarrow \text{CourseName}, \text{DeptName } \rightarrow \text{CourseName} \} \]?
Splitting and Combining FDs

- Can we split and combine left hand sides of FDs?
- For the relation Courses, is the FD
  \[\text{Number} \rightarrow \text{DeptName} \rightarrow \text{CourseName}\]
equivalent to the set of FDs
  \[
  \{\text{Number} \rightarrow \text{CourseName}, \text{DeptName} \rightarrow \text{CourseName}\}\]?
  NO!
Triviality of FDs

An FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is

▶ trivial if the $B$'s are a subset of the $A$'s, i.e.,
\[
\{ B_1, B_2, \ldots, B_m \} \subseteq \{ A_1, A_2, \ldots, A_n \}.
\]

▶ non-trivial if at least one $B$ is not among the $A$'s, i.e.,
\[
\{ B_1, B_2, \ldots, B_m \} - \{ A_1, A_2, \ldots, A_n \} \neq \emptyset.
\]

▶ completely non-trivial if none of the $B$'s are among the $A$'s, i.e.,
\[
\{ B_1, B_2, \ldots, B_m \} \cap \{ A_1, A_2, \ldots, A_n \} = \emptyset.
\]

▶ What good are trivial and non-trivial dependencies?

▶ Trivial dependencies are always true.

▶ They help simplify reasoning about FDs.

▶ Trivial dependency rule: The FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is equivalent to the FD $A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k$, where the $C$'s are those $B$'s that are not $A$'s, i.e.,
\[
\{ C_1, C_2, \ldots, C_k \} = \{ B_1, B_2, \ldots, B_m \} - \{ A_1, A_2, \ldots, A_n \}.
\]
Triviality of FDs

An FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is

- trivial if the $B$’s are a subset of the $A$’s,
Triviality of FDs

An FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is

- **trivial** if the $B$'s are a subset of the $A$'s, i.e.,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$.
Triviality of FDs

An FD \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \) is

- trivial if the \( B \)'s are a subset of the \( A \)'s, i.e.,
  \( \{ B_1, B_2, \ldots B_n \} \subseteq \{ A_1, A_2, \ldots A_n \} \).
- non-trivial if at least one \( B \) is not among the \( A \)'s,
Triviality of FDs

An FD $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ is

- trivial if the $B$’s are a subset of the $A$’s, i.e.,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$.

- non-trivial if at least one $B$ is not among the $A$’s, i.e.,
  $\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset$. 

Trivial dependency rule: The FD $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ is equivalent to the FD $A_1A_2 \ldots A_n \rightarrow C_1C_2 \ldots C_k$, where the $C$’s are those $B$’s that are not $A$’s, i.e.,

$\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} - \{A_1, A_2, \ldots, A_n\}$. 

T. M. Murali October 18, 25, 2010 CS 4604: Functional Dependencies
Triviality of FDs

An FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is

- **trivial** if the $B$’s are a subset of the $A$’s, i.e.,
  \[ \{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}. \]

- **non-trivial** if at least one $B$ is not among the $A$’s, i.e.,
  \[ \{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset. \]

- **completely non-trivial** if none of the $B$’s are among the $A$’s,
An FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is

- **trivial** if the $B$’s are a subset of the $A$’s, i.e.,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$.
- **non-trivial** if at least one $B$ is not among the $A$’s, i.e.,
  $\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset$.
- **completely non-trivial** if none of the $B$’s are among the $A$’s, i.e.,
  $\{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset$.
Triviality of FDs

An FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is

- **trivial** if the $B$’s are a subset of the $A$’s, i.e.,
  \[
  \{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}.
  \]

- **non-trivial** if at least one $B$ is not among the $A$’s, i.e.,
  \[
  \{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset.
  \]

- **completely non-trivial** if none of the $B$’s are among the $A$’s, i.e.,
  \[
  \{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset.
  \]

- What good are trivial and non-trivial dependencies?
  - Trivial dependencies are always true.
  - They help simplify reasoning about FDs.
Triviality of FDs

An FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is

- trivial if the $B$’s are a subset of the $A$’s, i.e.,
  \[\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}.\]

- non-trivial if at least one $B$ is not among the $A$’s, i.e.,
  \[\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset.\]

- completely non-trivial if none of the $B$’s are among the $A$’s, i.e.,
  \[\{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset.\]

What good are trivial and non-trivial dependencies?

- Trivial dependencies are always true.
- They help simplify reasoning about FDs.

- **Trivial dependency rule**: The FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is equivalent to the FD $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$, where the $C$’s are those $B$’s that are not $A$’s, i.e.,
  \[\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} - \{A_1, A_2, \ldots, A_n\}.\]
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E, \text{ and } F$ satisfies the FDs

\[ AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B \]

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?

\[ X = \{ A, B, C, D, E, F \} \text{, i.e., } AB \rightarrow ABCDEF \]

- $\{ B, C, F \}$ is a superkey.
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?\[X = \{A, B, C, D, E\}, \text{i.e., } AB \rightarrow ABCDE.\]
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  $$X = \{A, B, C, D, E\}, \text{ i.e., } AB \rightarrow ABCDE.$$ 

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?

$\{B, C, F\}$ is a superkey.
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  $$X = \{A, B, C, D, E\}, \text{ i.e., } AB \rightarrow ABCDE.$$  

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  $$Y = \{A, B, C, D, E, F\}, \text{ i.e., } BCF \rightarrow ABCDEF.$$
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

\[ AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B. \]

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  \[ X = \{ A, B, C, D, E \}, \text{ i.e., } AB \rightarrow ABCDE. \]

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  \[ Y = \{ A, B, C, D, E, F \}, \text{ i.e., } BCF \rightarrow ABCDEF \]

- what is the set $Z$ of attributes such that $AF \rightarrow Z$ is true?
Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  $$X = \{A, B, C, D, E\}, \text{ i.e., } AB \rightarrow ABCDE.$$  

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  $$Y = \{A, B, C, D, E, F\}, \text{ i.e., } BCF \rightarrow ABCDEF$$

- what is the set $Z$ of attributes such that $AF \rightarrow Z$ is true?
  $$Z = \{A, F\}, \text{ i.e., } AF \rightarrow AF.$$
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  $X = \{A, B, C, D, E\}$, i.e., $AB \rightarrow ABCDE$.

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  $Y = \{A, B, C, D, E, F\}$, i.e., $BCF \rightarrow ABCDEF$.

- what is the set $Z$ of attributes such that $AF \rightarrow Z$ is true?
  $Z = \{A, F\}$, i.e., $AF \rightarrow AF$.

- $X, Y, \text{ and } Z$ are the closures of $\{A, B\}, \{B, C, F\}, \text{ and } \{A, F\}$, respectively.
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

\[ AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B \]

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  \[ X = \{A, B, C, D, E\}, \text{ i.e., } AB \rightarrow ABCDE. \]
- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  \[ Y = \{A, B, C, D, E, F\}, \text{ i.e., } BCF \rightarrow ABCDEF \]
- what is the set $Z$ of attributes such that $AF \rightarrow Z$ is true?
  \[ Z = \{A, F\}, \text{ i.e., } AF \rightarrow AF. \]
- $X, Y,$ and $Z$ are the closures of $\{A, B\}$, $\{B, C, F\}$, and $\{A, F\}$, respectively.
- $\{B, C, F\}$ is a
Closures of Attributes: Example

Suppose a relation with attributes $A, B, C, D, E,$ and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  
  $X = \{A, B, C, D, E\}$, i.e., $AB \rightarrow ABCDE$.

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  
  $Y = \{A, B, C, D, E, F\}$, i.e., $BCF \rightarrow ABCDEF$.

- what is the set $Z$ of attributes such that $AF \rightarrow Z$ is true?
  
  $Z = \{A, F\}$, i.e., $AF \rightarrow AF$.

- $X, Y, \text{ and } Z$ are the closures of $\{A, B\}, \{B, C, F\}, \text{ and } \{A, F\}$, respectively.

- $\{B, C, F\}$ is a superkey.
Closures of Attributes: Definition

Given

▶ a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
▶ a set of FDs \( S \),

the \textit{closure} of \( \{A_1, A_2, \ldots, A_n\} \) under the FDs in \( S \) is

▶ the set of attributes \( \{B_1, B_2, \ldots, B_m\} \) such that for \( 1 \leq i \leq m \), the FD \( A_1A_2\ldots A_n \rightarrow B_i \) follows from \( S \).
▶ the closure is denoted by \( \{A_1, A_2, \ldots, A_n\}^+ \).
Closures of Attributes: Definition

Given

- a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
- a set of FDs \( S \),

the closure of \( \{A_1, A_2, \ldots, A_n\} \) under the FDs in \( S \) is

- the set of attributes \( \{B_1, B_2, \ldots, B_m\} \) such that for \( 1 \leq i \leq m \), the FD \( A_1A_2 \ldots A_n \rightarrow B_i \) follows from \( S \).
- the closure is denoted by \( \{A_1, A_2, \ldots, A_n\}^+ \).
- Which attributes must \( \{A_1, A_2, \ldots, A_n\}^+ \) contain at a minimum?
Closures of Attributes: Definition

Given

- a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
- a set of FDs \( S \),

the closure of \( \{A_1, A_2, \ldots, A_n\} \) under the FDs in \( S \) is

- the set of attributes \( \{B_1, B_2, \ldots, B_m\} \) such that for \( 1 \leq i \leq m \), the FD \( A_1A_2\ldots A_n \rightarrow B_i \) follows from \( S \).
- the closure is denoted by \( \{A_1, A_2, \ldots, A_n\}^+ \).
- Which attributes must \( \{A_1, A_2, \ldots, A_n\}^+ \) contain at a minimum? \( \{A_1, A_2, \ldots, A_n\} \). Why?
Closures of Attributes: Definition

Given

- a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
- a set of FDs \( S \),

the closure of \( \{A_1, A_2, \ldots, A_n\} \) under the FDs in \( S \) is

- the set of attributes \( \{B_1, B_2, \ldots, B_m\} \) such that for \( 1 \leq i \leq m \), the FD \( A_1 A_2 \ldots A_n \rightarrow B_i \) follows from \( S \).
- the closure is denoted by \( \{A_1, A_2, \ldots, A_n\}^+ \).
- Which attributes must \( \{A_1, A_2, \ldots, A_n\}^+ \) contain at a minimum? \( \{A_1, A_2, \ldots, A_n\} \). Why? \( A_1 A_2 \ldots A_n \rightarrow A_i \) is a trivial FD.
Given

- a set of attributes \(\{A_1, A_2, \ldots, A_n\}\) and
- a set of FDs \(S\),
- compute \(X = \{A_1, A_2, \ldots, A_n\}^+\).
Closures of Attributes: Algorithm

Given

- a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
- a set of FDs \( S \),
- compute \( X = \{A_1, A_2, \ldots, A_n\}^+ \).

1. Use the splitting rule so that each FD in \( S \) has one attribute on the right.
2. Set \( X \leftarrow \{A_1, A_2, \ldots, A_n\} \).
3. Find an FD \( B_1 B_2 \ldots B_k \rightarrow C \) in \( S \) such that \( \{B_1, B_2, \ldots B_k\} \subseteq X \) but \( C \not\in X \).
4. Add \( C \) to \( X \).
5. Repeat the last two steps until you cannot find such an attribute \( C \).
6. The final value of \( X \) is the desired closure.
Closures of Attributes: Algorithm

Given
- a set of attributes \{A_1, A_2, \ldots, A_n\} and
- a set of FDs \(S\),
- compute \(X = \{A_1, A_2, \ldots, A_n\}^+\).

1. Use the splitting rule so that each FD in \(S\) has one attribute on the right.
2. Set \(X \leftarrow \{A_1, A_2, \ldots, A_n\}\).
3. Find an FD \(B_1 B_2 \ldots B_k \rightarrow C\) in \(S\) such that \(\{B_1, B_2, \ldots B_k\} \subseteq X\) but \(C \notin X\).
4. Add \(C\) to \(X\).
5. Repeat the last two steps until you cannot find such an attribute \(C\).
6. The final value of \(X\) is the desired closure.

- Why does the algorithm compute the closure correctly? Read Chapter 3.2.5 of the textbook.
Why is the Concept of Closures Useful?

1. Prove correctness of rules for manipulating FDs.
Why is the Concept of Closures Useful?

1. Prove correctness of rules for manipulating FDs. For example,
   - **Transitive rule:** if
     
     \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \]
     and
     
     \[ B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \]
     then
     
     \[ A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k \].
Why is the Concept of Closures Useful?

1. Prove correctness of rules for manipulating FDs. For example,
   - **Transitive rule**: if
     \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \]
     
     and
     
     \[ B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \]
     
     then
     
     \[ A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k. \]
   - To prove this rule, simply check if
     \[ \{C_1, C_2, \ldots, C_k\} \subseteq \{A_1, A_2, \ldots, A_n\}^+. \]
Why is the Concept of Closures Useful?

1. Prove correctness of rules for manipulating FDs. For example,
   - **Transitive rule:** if
     \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \]
     and
     \[ B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \]
     then
     \[ A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k. \]
   - To prove this rule, simply check if
     \[ \{C_1, C_2, \ldots, C_k\} \subseteq \{A_1, A_2, \ldots, A_n\}^+. \]

2. Check if a “new” FD \( A_1 A_2 \ldots A_n \rightarrow B \) follows from a set of FDs \( S \):
   simply check if \( B \) is in \( \{A_1, A_2, \ldots, A_n\}^+ \) under \( S \).
Why is the Concept of Closures Useful?

1. Prove correctness of rules for manipulating FDs. For example,
   - **Transitive rule**: if
     \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \]
     and
     \[ B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \]
     then
     \[ A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k . \]
   - To prove this rule, simply check if
     \[ \{ C_1, C_2, \ldots, C_k \} \subseteq \{ A_1, A_2, \ldots, A_n \}^+ . \]

2. Check if a “new” FD \( A_1 A_2 \ldots A_n \rightarrow B \) follows from a set of FDs \( S \): simply check if \( B \) is in \( \{ A_1, A_2, \ldots A_n \}^+ \) under \( S \).

3. Procedurally define keys.
Why is the Concept of Closures Useful?

1. Prove correctness of rules for manipulating FDs. For example,
   - **Transitive rule**: if
     \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \]
     and
     \[ B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \]
     then
     \[ A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k. \]
   - To prove this rule, simply check if
     \[ \{C_1, C_2, \ldots, C_k\} \subseteq \{A_1, A_2, \ldots, A_n\}^+. \]

2. Check if a “new” FD \( A_1 A_2 \ldots A_n \rightarrow B \) follows from a set of FDs \( S \): simply check if \( B \) is in \( \{A_1, A_2, \ldots, A_n\}^+ \) under \( S \).

3. Procedurally define keys. A set of attributes \( X \) is a key for a relation \( R \) if and only if
   - \( \{X\}^+ \) is the set of all attributes of \( R \) and
   - for no attribute \( A \in X \) is \( \{X - \{A\}\}^+ \) the set of all attributes of \( R \).
Examples of Closure Computations

Consider the “bad” relation $\text{Students}(\text{Id}, \text{Name}, \text{AdvisorId}, \text{AdvisorName}, \text{FavouriteAdvisorId})$.

What are the FDs that hold in this relation?

- $\text{Id} \rightarrow \text{Name}$
- $\text{Id} \rightarrow \text{FavouriteAdvisorId}$
- $\text{AdvisorId} \rightarrow \text{AdvisorName}$
Examples of Closure Computations

Consider the “bad” relation Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId).

What are the FDs that hold in this relation?

- $Id \rightarrow Name$
- $Id \rightarrow FavouriteAdvisorId$
- $AdvisorId \rightarrow AdvisorName$

To compute the key for this relation,

1. Compute the closures for all sets of attributes.
2. Find the minimal set of attributes whose closure is the set of all attributes.
Algorithm for Computing Keys

Given

- a relation $R(A_1, A_2, \ldots, A_n)$ and
- the set of all FDs $S$ that hold in $R$,
- compute all the keys of $R$. 

Running time of the algorithm is exponential in the number of attributes.
Algorithm for Computing Keys

Given

- a relation \( R(A_1, A_2, \ldots, A_n) \) and
- the set of all FDs \( S \) that hold in \( R \),
- compute all the keys of \( R \).

1. For every subset \( K \subseteq \{A_1, A_2, \ldots, A_n\} \), compute \( \{K\}^+ \).
2. If \( \{K\}^+ = \{A_1, A_2, \ldots, A_n\} \) and for every attribute \( A \), \( \{K - \{A\}\}^+ \neq \{A_1, A_2, \ldots, A_n\} \), then output \( K \) as a key.
Algorithm for Computing Keys

Given

- a relation $R(A_1, A_2, \ldots, A_n)$ and
- the set of all FDs $S$ that hold in $R$,
- compute all the keys of $R$.

1. For every subset $K \subseteq \{A_1, A_2, \ldots, A_n\}$, compute $\{K\}^+$.
2. If $\{K\}^+ = \{A_1, A_2, \ldots, A_n\}$ and for every attribute $A$, $\{K - \{A\}\}^+ \neq \{A_1, A_2, \ldots, A_n\}$, then output $K$ as a key.

- Running time of the algorithm is exponential in the number of attributes.
Armstrong’s Axioms

- We can use closures of attributes to determine if any FD follows from a given set of FDs.
- Armstrong’s axioms: complete set of inference rules from which it is possible to derive every FD that follows from a given set:
Armstrong’s Axioms

▶ We can use closures of attributes to determine if any FD follows from a given set of FDs.

▶ Armstrong’s axioms: complete set of inference rules from which it is possible to derive every FD that follows from a given set:

Reflexivity If \( \{ B_1, B_2, \ldots, B_m \} \subseteq \{ A_1, A_2, \ldots, A_n \} \), then \( A_1A_2\ldots A_m \rightarrow B_1B_2\ldots B_m \) (Trivial FDs).
Armstrong’s Axioms

- We can use closures of attributes to determine if any FD follows from a given set of FDs.

- Armstrong’s axioms: complete set of inference rules from which it is possible to derive every FD that follows from a given set:
  
  **Reflexivity**  
  If \( \{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\} \), then  
  \[ A_1A_2\ldots A_m \rightarrow B_1B_2\ldots B_m \]  
  (Trivial FDs).

  **Augmentation**  
  If \( A_1A_2\ldots A_m \rightarrow B_1B_2\ldots B_m \), then  
  \[ A_1A_2\ldots A_mC_1C_2\ldots C_k \rightarrow B_1B_2\ldots B_mC_1C_2\ldots C_k \], for any set of attributes \( \{C_1, C_2, \ldots, C_k\} \).
We can use closures of attributes to determine if any FD follows from a given set of FDs.

Armstrong’s axioms: complete set of inference rules from which it is possible to derive every FD that follows from a given set:

**Reflexivity** If \( \{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\} \), then \( A_1 A_2 \ldots A_m \rightarrow B_1 B_2 \ldots B_m \) (Trivial FDs).

**Augmentation** If \( A_1 A_2 \ldots A_m \rightarrow B_1 B_2 \ldots B_m \), then \( A_1 A_2 \ldots A_m C_1 C_2 \ldots C_k \rightarrow B_1 B_2 \ldots B_m C_1 C_2 \ldots C_k \), for any set of attributes \( \{C_1, C_2, \ldots, C_k\} \).

**Transitivity** If \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \) and \( B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \) then \( A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k \).
Differences in Notation

Relation schema \( R(A_1, A_2, A_3) \): parentheses surround attributes, attributes separated by commas.

Set of attributes \( \{A_1, A_2, A_3\} \): curly braces surround attributes, attributes separated by commas.

FD \( A_1 A_2 \rightarrow A_3 \): no parentheses or curly braces, attributes separated by spaces, arrows separates left hand side and right hand side.

Set of FDs \( \{A_1 A_2 \rightarrow A_3, A_2 \rightarrow A_1\} \): curly braces surround FDs, FDs separated by commas.
A relation may have a large set of equivalent sets of FDs.

If we are given a set $S$ of FDs, then any set of FDs that is equivalent to $S$ is called a *basis* of $S$.
A relation may have a large set of equivalent sets of FDs.

If we are given a set $S$ of FDs, then any set of FDs that is equivalent to $S$ is called a basis of $S$.

We will limit ourselves to bases in which each FD has only one attribute on the right hand side.
A relation may have a large set of equivalent sets of FDs.

If we are given a set $S$ of FDs, then any set of FDs that is equivalent to $S$ is called a \textit{basis} of $S$.

We will limit ourselves to bases in which each FD has only one attribute on the right hand side.

A set $B$ of FDs is a \textit{minimal basis} for a relation $R$ if

1. Every FD in $B$ has one attribute on the right hand side.
2. If we remove any FD from $B$, then the result is not a basis.
3. If for any FD in $B$, we remove one or more attributes from the left hand side of the FD, then the result is not a basis.
Example of Minimal Basis

- \( R(A, B, C) \) is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in \( R \) and what are the minimal bases? (Assume only one attribute on the right-hand side, only non-trivial FDs)

FDs:
- \( A \rightarrow B \), \( A \rightarrow C \), \( B \rightarrow A \), \( B \rightarrow C \), \( C \rightarrow A \), \( C \rightarrow B \), \( AB \rightarrow C \), \( BC \rightarrow A \), \( AC \rightarrow B \).

Minimal bases:
- \{ \( A \rightarrow B \), \( B \rightarrow A \), \( B \rightarrow C \), \( C \rightarrow B \) \}, and others.
Example of Minimal Basis

- \( R(A, B, C) \) is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in \( R \) and what are the minimal bases? (Assume only one attribute on the right-hand side, only non-trivial FDs)
- FDs:

\[
\begin{align*}
A & \rightarrow B, \\
B & \rightarrow A, \\
B & \rightarrow C, \\
C & \rightarrow B
\end{align*}
\]
Example of Minimal Basis

- \( R(A, B, C) \) is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in \( R \) and what are the minimal bases? (Assume only one attribute on the right-hand side, only non-trivial FDs)
- FDs: \( A \rightarrow B, \ A \rightarrow C, \ B \rightarrow A, \ B \rightarrow C, \ C \rightarrow A, \ C \rightarrow B \),
Example of Minimal Basis

- $R(A, B, C)$ is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in $R$ and what are the minimal bases? (Assume only one attribute on the right-hand side, only non-trivial FDs)
- FDs: $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow A$, $B \rightarrow C$, $C \rightarrow A$, $C \rightarrow B$, $AB \rightarrow C$, $BC \rightarrow A$, $AC \rightarrow B$. 
Example of Minimal Basis

- \( R(A, B, C) \) is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in \( R \) and what are the minimal bases? (Assume only one attribute on the right-hand side, only non-trivial FDs)
- FDs: \( A \rightarrow B \), \( A \rightarrow C \), \( B \rightarrow A \), \( B \rightarrow C \), \( C \rightarrow A \), \( C \rightarrow B \), \( AB \rightarrow C \), \( BC \rightarrow A \), \( AC \rightarrow B \).
- Minimal bases:
Example of Minimal Basis

- $R(A, B, C)$ is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in $R$ and what are the minimal bases? (Assume only one attribute on the right-hand side, only non-trivial FDs)
- FDs: $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow A$, $B \rightarrow C$, $C \rightarrow A$, $C \rightarrow B$, $AB \rightarrow C$, $BC \rightarrow A$, $AC \rightarrow B$.
- Minimal bases: \{$A \rightarrow B$, $B \rightarrow A$, $B \rightarrow C$, $C \rightarrow B$\}, \{$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$\}, and others.