SC2207/CZ2007 Introduction to Database Systems (Week 3)

Topic 2: Functional Dependencies (2)

This Lecture

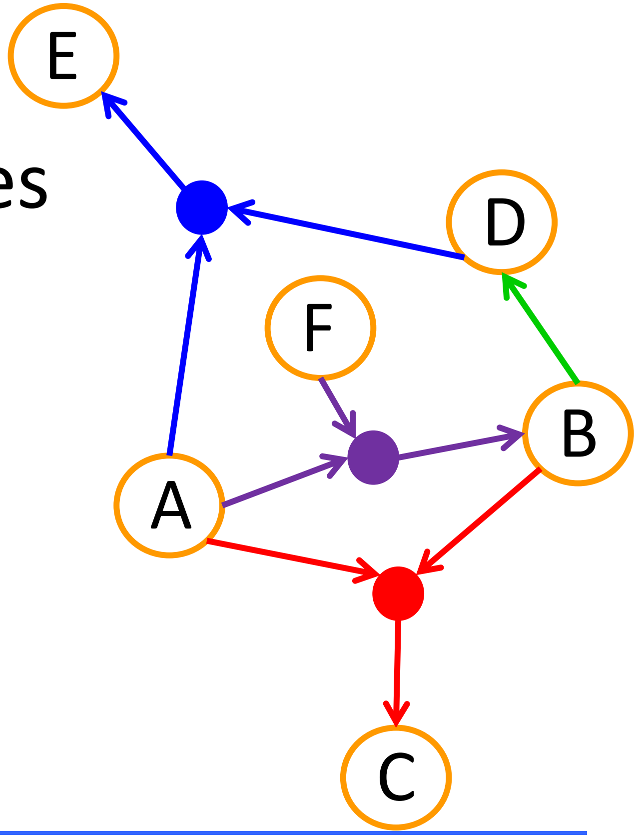
- Closure
- Keys
- Finding keys
- Normal forms

Closure

- Let $S = \{A_1, A_2, \ldots, A_n\}$ be a set S of n attributes
- Closure of S is the set of attributes that can be determined by $A_1, A_2, \ldots, A_n$
- Notation: $\{A_1, A_2, \ldots, A_n\}^+$
- Example
  - Given $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$
  - $\{A\}^+ = \{A, B, C, D, E\}$
  - $\{B\}^+ = \{B, C, D, E\}$
  - $\{D\}^+ = \{D, E\}$
  - $\{E\}^+ = \{E\}$

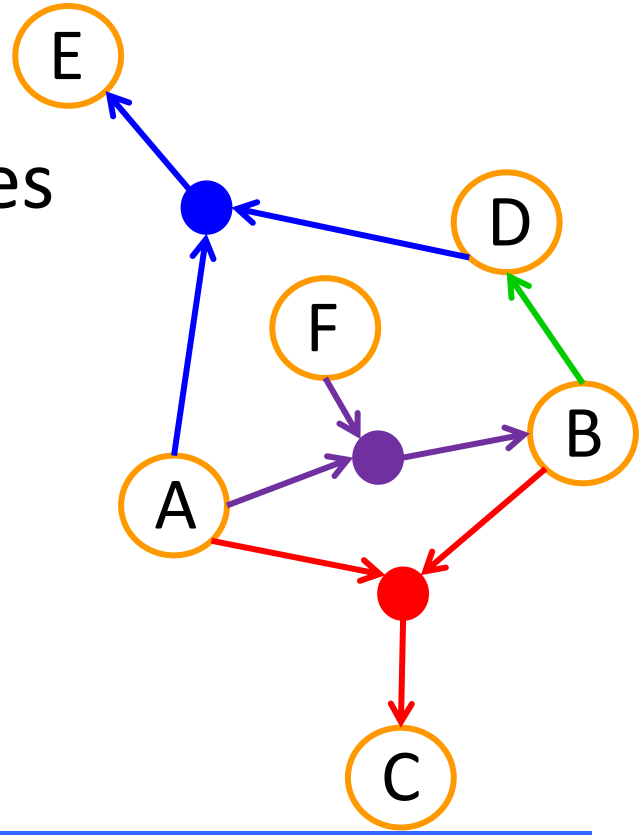
Exercise

- A table with six attributes A, B, C, D, E, F
- AB→C, AD→E, B→D, AF→B
- Compute the following closures
  - $\{B, C\}^+ =$
  - $\{A, B\}^+ =$
  - $\{A, F\}^+ =$



Exercise

- A table with six attributes A, B, C, D, E, F
- $AB \rightarrow C$, $AD \rightarrow E$, $B \rightarrow D$, $AF \rightarrow B$
- Compute the following closures
  - $\{B, C\}^+ = \{B, C, D\}$
  - $\{A, B\}^+ = \{A, B, C, D, E\}$
  - $\{A, F\}^+ = \{A, B, C, D, E, F\}$



Closure & FD

- To prove that $X \rightarrow Y$ holds, we only need to show that $\{X\}^+$ contains Y
- $AB \rightarrow C$, $AD \rightarrow E$, $B \rightarrow D$, $AF \rightarrow B$
- Prove that $AF \rightarrow D$
- $\{AF\}^+ = \{AFBCDE\}$, which contains D
- Therefore, $AF \rightarrow D$ holds

Closure & FD

- To prove that X → Y does not hold, we only need to show that $\{X\}^+$ does not contain Y
- AB→C, AD→E, B→D, AF→B
- Prove that AD→F does not hold
- $\{AD\}^+ = \{ADE\}$, which does not contain F
- Therefore, AD→F does not hold

This Lecture

- Closure
- Keys
- Finding keys
- Normal forms

Roadmap

- Now we know what a closure is
- We discuss the usage of closure for finding the keys in a table
  - This is a necessary step for checking whether a table is good or not
- Concepts
  - Superkeys
  - Keys
  - Candidate keys
  - Primary keys / Secondary keys

Notion of keys in a table

$\neq$

keys in a entity set

Superkeys of a Table

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

- Definition: A set of attributes in a table that determines all other attributes
- Example:
  - {NRIC} is a superkey (happens to be minimal)
  - Since NRIC → Name, Postal, Address
  - {NRIC, Name} is a superkey (not minimal)
  - Since {NRIC, Name} → Postal, Address

Keys of a Table

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

- Definition: A superkey that is minimal
- i.e., if we remove any attribute from the superkey, it will not be a superkey anymore
- Example:
  - {NRIC} is a superkey (happens to be minimal)
  - Since NRIC → Name, Postal, Address
  - {NRIC, Name} is a superkey
  - Since {NRIC, Name} → Postal, Address
  - NRIC is a key, but {NRIC, Name} is not a key, but a superkey

Keys of a Table

Name	NRIC	Postal	Address
Alice	1234	939450	Jurong East
Bob	5678	234122	Pasir Ris
Cathy	3576	420923	Yishun

- Note: Not to be confused with the keys of entity sets because keys in entity sets need not be minimal.

Candidate Keys

Name	NRIC	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3576	3	420923	Yishun

- A table may have multiple keys
- In that case, each key is referred to as a candidate key
- Example:
  - {NRIC} is a key
  - Since NRIC → Name, StudentID, Postal, Address
  - {StudentID} is a key
  - Since StudentID → Name, NRIC, Postal, Address
  - Both {NRIC} and {StudentID} are candidate keys

Primary and Secondary Keys

Name	NRIC	StudentID	Postal	Address
Alice	1234	1	939450	Jurong East
Bob	5678	2	234122	Pasir Ris
Cathy	3576	3	420923	Yishun

- When a table have multiple keys …
- We choose one of them as the primary key
- The others are referred to as secondary keys
- Example:
  - {NRIC} is a key
  - {StudentID} is a key
  - If we choose {NRIC} as the primary key
  - Then {StudentID} is the secondary key

Summary

- Superkeys
  - A set of attributes that determines all other attributes in a table
- Keys
  - A minimal set of attributes that determines all other attributes in a table
- Example
  - R(A, B, C, D)
  - Given: A→BCD, BC→A
  - A is a key and a superkey
  - BC is also a key and a superkey
  - AB is not a key; it is only a superkey

This Lecture

- Closure
- Keys
- Finding keys
- Normal forms

Finding the Keys

- To check whether a table is “good”, we need to find the keys of the table
- How do we do that?
- Use functional dependencies (FDs) and closures

Example

- Definition of a Key: A minimal set of attributes that determines all other attributes
- A table R(A, B, C)
- FDs given: $A \rightarrow B$, $B \rightarrow C$
- Is A a key?
- $\{A\}^+ = \{ABC\}$, i.e., $A \rightarrow ABC$. Yes.
- Is B a key?
- $\{B\}^+ = \{BC\}$, i.e., B does not decide A. No.
- Is C a key?
- $\{C\}^+ = \{C\}$. No.
- Is AB a key?
- No, since A is already a key.
- What about BC, AC, ABC?

Example

- Definition of a Key: A minimal set of attributes that determines all other attributes
- A table R(A, B, C)
- FDs given: $A \rightarrow B$
- Is A a key?
- $\{A\}^+ = \{A, B\}$. No.
- Is B or C a key?
- $\{B\}^+ = \{B\}$. $\{C\}^+ = \{C\}$. No.
- Is AB or BC a key?
- $\{AB\}^+ = \{AB\}$. $\{BC\}^+ = \{BC\}$. No.
- Is AC a key?
- $\{AC\}^+ = \{\underline{ABC}\}$. Yes.
- Is ABC a key?

Finding the Keys: Algorithm

- Check all possible combinations of attributes in the table
  - Example: A, B, C, AB, BC, AC, ABC
- For each combination, compute its closure
  - Example: $\{A\}^+ = \ldots$, $\{B\}^+ = \ldots$, $\{C\}^+ = \ldots$, …
- If a closure contains ALL attributes, then the combination might be a key (or superkey)
  - Example: $\{A\}^+ = \{ABC\}$
- Make sure that you select only keys
  - Example: $\{A\}^+ = \{ABC\}$, $\{AB\}^+ = \{ABC\}$, don't select AB

Example

- A table R(A, B, C, D)
- AB→C, AD→B, B→D
- First, enumerate all attribute combinations:
  - {A}, {B}, {C}, {D}
  - {AB}, {AC}, {AD}, {BC}, {BD}, {CD}
  - {ABC}, {ABD}, {ACD}, {BCD}
  - {ABCD}

Example

- A table R(A, B, C, D)
- AB→C, AD→B, B→D
- Second, compute the closures:
  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{BD\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
  - $\{AB\}^+ = \{\underline{ABCD}\}$, $\{AC\}^+ = \{AC\}$, $\{AD\}^+ = \{\underline{ABCD}\}$
  - $\{BC\}^+ = \{BCD\}$, $\{BD\}^+ = \{BD\}$, $\{CD\}^+ = \{CD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{ACD\}^+ = \{\underline{ABCD}\}$
  - $\{BCD\}^+ = \{BCD\}$
  - $\{ABCD\}^+ = \{\underline{ABCD}\}$
- Finally, output the keys

A Small Trick

- Always check small combinations first
- A table R(A, B, C, D)
- $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$
- Compute the closures:
  - $\{A\}^+ = \{ABCD\}$, $\{B\}^+ = \{ABCD\}$, $\{C\}^+ = \{ABCD\}$, $\{D\}^+ = \{ABCD\}$
  - No need to check others
  - The others are all superkeys but not keys
- Keys: {A}, {B}, {C}, {D}

Another Small Trick

- A table R(A, B, C, D)
- AB→C, AD→B, B→D
- Notice that A does not appear at the right hand side of any functional dependencies
- In that case, A must be in every key
- Keys of R: AB, AD (From the previous slide)
- In general, if an attribute that does not appear in the right hand side of any FD, then it must be in every key

Exercise (Find the Keys)

- A table R(A, B, C, D)
- $A \rightarrow B$, $A \rightarrow C$, $C \rightarrow D$
- A must be in every key
- Compute the closures:
  - $\{A\}^+ = \{\underline{ABCD}\}$
  - No need to check others
- Keys: {A}

Exercise (Find the Keys)

- A table R(A, B, C, D, E)
- $AB \rightarrow C$, $C \rightarrow B$, $BC \rightarrow D$, $CD \rightarrow E$
- A must be in every key
- Compute the closures:
  - $\{A\}^+ = \{A\}$
  - $\{AB\}^+ = \{\underline{ABCDE}\}$
  - $\{AC\}^+ = \{\underline{ACBDE}\}$
  - $\{AD\}^+ = \{AD\}$, $\{AE\}^+ = \{AE\}$
  - $\{ADE\}^+ = \{ADE\}$

Exercise (Find the Keys)

- A table R(A, B, C, D, E, F)
- $AB \rightarrow C$, $C \rightarrow B$, $CBE \rightarrow D$, $D \rightarrow EF$
- A must be in every key
- Compute the closures:
  - $\{A\}^+ = \{A\}$
  - $\{AB\}^+ = \{ABC\}$
  - $\{AC\}^+ = \{ACB\}$
  - $\{AD\}^+ = \{ADEF\}$
  - $\{AE\}^+ = \{AE\}$, $\{AF\}^+ = \{AF\}$
  - $\{ABC\}^+ = \{ABC\}$
  - $\{ABD\}^+ = \{ABE\}^+ = \{ACD\}^+ = \{ACE\}^+ = \{\underline{ABCDEF}\}$
  - $\{ADE\}^+ = \{ADEF\}$

This Lecture

- Closure
- Keys
- Finding keys
- Normal forms

Recap

- We have talked about a lot of abstract stuff
  - Functional Dependencies
  - Armstrong’s Axioms
  - Closures
  - Keys
- Again, what are we doing here?
- The above stuff is needed for normal forms and normalization

Recap

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- The table has a number of anomalies
  - Redundancy, insertion anomalies, etc.
- We would like to get rid of tables like this
- For this purpose, we need
  - A method to detect "bad tables" like this
  - A method to fix "bad tables" like this

Recap

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83848384	Jurong East
Bob	5678	98765432	Pasir Ris

- The table has a number of anomalies
  - Redundancy, insertion anomalies, etc.
- We would like to get rid of tables like this
- For this purpose, we need
  - A method to detect “bad tables” like this: normal forms
  - A method to fix “bad tables” like this: normalization

Normal Forms

- Some conditions that a “good” table must satisfy
- Intuition
  - A student can get first-class honor, if her GPA is at least 4.0
  - “At least 4.0” is a condition that first-class-honor students must satisfy
  - Normal forms are conditions for “good” tables

Normal Forms

- Various normal forms (in increasing order of strictness)
  - First normal form
  - Second normal form
  - Third normal form (3NF)
  - Boyce-Codd normal form (BCNF)
  - Fourth normal form
  - Fifth normal form
  - Sixth normal form
- 3NF and BCNF are most commonly used

Normal Forms

- Various normal forms (in increasing order of strictness)
  - First normal form
  - Second normal form
  - Third normal form (3NF)
  - Boyce-Codd normal form (BCNF)
  - Fourth normal form
  - Fifth normal form
  - Sixth normal form
- 3NF and BCNF are most commonly used

Next lecture:

Topic 3: Boyce-Codd Normal Form (1)