Definition of Candidate Keys Based on FDs

- Definition by means of FDs
- A set $A \subseteq R$ is a candidate key of a relation schema R if
  - [Uniqueness] $A \rightarrow R - A$

    A functionally determines all the other attributes in R

    Since A functionally determines itself, we obtain: $A \rightarrow R$
  - [Minimality] There is no proper subset $C \subset A$ so that $C \rightarrow R$ holds

Checking the Validity of a Functional Dependency

- Alternative characterization of an FD $A \rightarrow B$ by a Relational Algebra expression
  - Let $A = \{A_1, ..., A_n\}$, and let $dom(A) = dom(A_1) \times ... \times dom(A_n)$
  - Let $A = v$ stand for $A_1 = v_1 \wedge ... \wedge A_n = v_n$
  - The FD $A \rightarrow B$ holds on R if $\forall\ v \in dom(A) : |\pi_B(\sigma_{A=v}(R))| \leq 1$

- This leads to a simple algorithm which computes whether a given relation R satisfies a given FD $A \rightarrow B$:

bool *FDIsValidOnRelation*(R, $A \rightarrow B$)
// Input: Relation R and FD $A \rightarrow B$
// Output: *true*, if $A \rightarrow B$ holds on R; *false* otherwise
Sort R with respect to A-values
if all groups consisting of tuples with equal A-values
    also have equal B-values
  then return *true*
  else return *false*

Closure of a Set of Functional Dependencies (I)

- Goal: Compute all logically implied or inferred FDs for a given set F of FDs
- An FD f on a relation schema R is logically implied by a set F of FDs on R if every relation r_R that satisfies F also satisfies f
- Example
  - We introduce the notation ABC (i.e., juxtaposition) for the set $\{A, B, C\}$ of attributes; then A represents $\{A\}$
  - The context decides whether, e.g., A is the name of an attribute or the name of an attribute set
  - If A, B, and C are attributes, then $AC \rightarrow BC$ means $\{A, C\} \rightarrow \{B, C\}$, and $A \rightarrow B$ means $\{A\} \rightarrow \{B\}$
  - Let $R(A, B, C, G, H, I)$ be a relation schema, and let $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$ be a set of FDs defined on R
  - Statement: The FD $A \rightarrow H$ is logically implied
  - We show: Whenever a relation satisfies F, $A \rightarrow H$ must also be satisfied by that relation
  - Suppose that t_1 and t_2 are tuples such that $t_1[A] = t_2[A]$ holds

Closure of a Set of Functional Dependencies (II)

- Example (*continued*)
  - Since we know that $A \rightarrow B$ holds, it follows from the definition of FDs that $t_1[B] = t_2[B]$ holds
  - Since we know that $B \rightarrow H$ holds, it follows from the definition of FDs that $t_1[H] = t_2[H]$ holds
  - We have shown that whenever t_1 and t_2 are tuples such that $t_1[A] = t_2[A]$ holds, it must be that $t_1[H] = t_2[H]$ holds
  - But this is exactly the definition of $A \rightarrow H$
  - We see that $A \rightarrow H$ is a transitive FD

- Given a set F of FDs, the closure of F, denoted by F^+, is the set of all FDs that can be logically implied by the FDs in F

- Armstrong's axioms are inference rules that provide a simpler and higher-level technique for reasoning about FDs than deploying the formal definition of FDs

Closure of a Set of Functional Dependencies (III)

- Given a relation schema R, a set F of FDs on R, and A, B, $C \subseteq R$, the following inference rules are used to compute F^+ (Armstrong's axioms):
  - Reflexivity rule: Let $B \subseteq A$. Then $A \rightarrow B$ (special case: $A \rightarrow A$) holds.
  - Augmentation rule: If $A \rightarrow B$ holds, then $A \cup C \rightarrow B \cup C$ holds.
  - Transitivity rule: If $A \rightarrow B$ and $B \rightarrow C$ holds, then $A \rightarrow C$ holds.
- It can be formally shown that these rules are sound and complete
  - Soundness: No incorrect FDs are generated, and the inferred FDs hold for all relations of this schema
  - Completeness: All valid FDs in F^+ can be logically implied by these rules
- Although Armstrong's axioms are complete, it is convenient to add three further derived inference rules:
  - Union rule: If $A \rightarrow B$ and $A \rightarrow C$ holds, then $A \rightarrow B \cup C$ holds.
  - Decomposition rule: If $A \rightarrow B \cup C$ holds, then $A \rightarrow B$ and $A \rightarrow C$ holds.
  - Pseudotransitivity rule: If $A \rightarrow B$ and $B \cup C \rightarrow D$ holds, then $A \cup C \rightarrow D$ holds.

Closure of a Set of Functional Dependencies (IV)

- Example
  - Schema supplier(<u>sname</u>, saddr, <u>product</u>, price)
  - Valid FDs, e.g.: {sname} $\rightarrow$ {saddr}, {sname, product} $\rightarrow$ {price}, {sname} $\rightarrow$ {sname}, {sname, product} $\rightarrow$ {product}
  - It is to be shown: {sname, product} $\rightarrow$ {saddr} is also satisfied.
    - We have: {sname} $\rightarrow$ {saddr}
    - Applying the augmentation rule:

      {sname, product} $\rightarrow$ {saddr, product}.
    - Applying the decomposition rule: {sname, product} $\rightarrow$ {saddr}
- Example
  - Given the schema $R(A, B, C, D)$ and $F = \{A \rightarrow B, C \rightarrow D\}$ on R, show that $AC \rightarrow BD$ holds
  - Applying the augmentation rule: $A \rightarrow B \Rightarrow AC \rightarrow BC$
  - Applying the augmentation rule: $C \rightarrow D \Rightarrow BC \rightarrow BD$
  - Applying the transitivity rule: $AC \rightarrow BC \wedge BC \rightarrow BD \Rightarrow AC \rightarrow BD$

Closure of a Set of Functional Dependencies (V)

- Algorithm to compute the closure F^+, given F

  F^+ *CalculateFDClosure*(F)

  // Input: Set F of FDs

  // Output: Closure F^+ of F

  $F^+ = F$

  repeat

      for each functional dependency f in F^+ **do**

          Apply reflexivity and augmentation rules to F^+

          Add the resulting functional dependencies to F^+

      for each pair of functional dependencies f_1 and f_2 in F^+ **do**

          if f_1 and f_2 can be combined using transitivity **then**

              Add the resulting functional dependency to F^+

  until F^+ does not change any further

  return F^+

Closure of a Set of Functional Dependencies (VI)

- By applying Armstrong's axioms repeatedly (see algorithm above), we can find all FDs of F^+, given F

- Problems of the algorithm
  - An FD to be added to the current F^+ may already be present so that F^+ remains unchanged
  - Algorithm has exponential run time complexity and is hence very slow

- Example
  - Given the schema $R(A, B, C)$ and the set $F = \{A \rightarrow B, B \rightarrow C\}$ on R, determine the closure F^+
  - Start: $A \rightarrow B$, $B \rightarrow C$
  - Reflexivity rule (Round 1)
    - $A \rightarrow A$, $B \rightarrow B$, $C \rightarrow C$

Closure of a Set of Functional Dependencies (VII)

- Example (*continued*)
  - Augmentation rule (Round 1)
    - Augmentation with A: $A \rightarrow AB$, $AB \rightarrow AC$
    - Augmentation with B: $AB \rightarrow B$, $B \rightarrow BC$
    - Augmentation with C: $AC \rightarrow BC$, $BC \rightarrow C$
    - Augmentation with AB: $AB \rightarrow AB$, $AB \rightarrow ABC$
    - Augmentation with AC: $AC \rightarrow ABC$, $ABC \rightarrow AC$
    - Augmentation with BC: $ABC \rightarrow BC$, $BC \rightarrow BC$
    - Augmentation with ABC: $ABC \rightarrow ABC$
  - Transitivity rule (Round 1)
    - $A \rightarrow B \wedge B \rightarrow C \Rightarrow A \rightarrow C$
    - $A \rightarrow B \wedge B \rightarrow BC \Rightarrow A \rightarrow BC$
    - $A \rightarrow AB \wedge AB \rightarrow AC \Rightarrow A \rightarrow AC$
    - $A \rightarrow AB \wedge AB \rightarrow ABC \Rightarrow A \rightarrow ABC$
    - $AB \rightarrow AC \wedge AC \rightarrow BC \Rightarrow AB \rightarrow BC$

Closure of a Set of Functional Dependencies (VIII)

- Example (*continued*)
  - Transitivity rule (Round 1) (*continued*)
    - $AB \to B \wedge B \to C \Rightarrow AB \to C$
    - $AC \to B \wedge B \to C \Rightarrow AC \to C$
    - $AC \to ABC \wedge ABC \to AC \Rightarrow AC \to AC$
    - $ABC \to BC \wedge BC \to C \Rightarrow ABC \to C$
  - Reflexivity rule (Round 2)
    - $AB \to AC \Rightarrow AB \to A$
    - $AC \to BC \Rightarrow AC \to B$
    - $ABC \to AC \Rightarrow ABC \to A$
    - $ABC \to BC \Rightarrow ABC \to B$
    - $BC \to BC \Rightarrow BC \to B$
    - $ABC \to ABC \Rightarrow ABC \to AB$
    - $ABC \to ABC \Rightarrow ABC \to AC$
    - $AC \to AC \Rightarrow AC \to A$

Closure of a Set of Functional Dependencies (IX)

- Example (*continued*)
  - Augmentation rule (Round 2)
    - No new FDs
  - Transitivity rule (Round 2)
    - No new FDs
  - Reflectivity rule (Round 3)
    - No new FDs
  - Augmentation rule (Round 3)
    - No new FDs
  - Transitivity rule (Round 3)
    - No new FDs
  - Algorithm terminates with 35 found FDs

Closure of a Set of Functional Dependencies (X)

- Example (*continued*)
  - $F^+ = \{$

$$A \to A,\ A \to B,\ A \to C,\ A \to BC,\ A \to AB,\ A \to AC,\ A \to ABC,$$
$$B \to B,\ B \to C,\ B \to BC,\ C \to C,\ AB \to A,\ AB \to B,\ AB \to C,$$
$$AB \to AB,\ AB \to BC,\ AB \to AC,\ AB \to ABC,\ BC \to B,\ BC \to C,$$
$$BC \to BC,\ AC \to A\ ,\ AC \to B,\ AC \to C,\ AC \to AC,\ AC \to AB,$$
$$AC \to BC,\ AC \to ABC,\ ABC \to A,\ ABC \to B,\ ABC \to C,$$
$$ABC \to BC,\ ABC \to AB,\ ABC \to AC,\ ABC \to ABC$$

$\}$

  - We learn:
    - Manual computation of F^+ is very error-prone
    - Many FDs are generated again and again (redundant work)
    - Many searches needed to check if a new FD has already been found before
    - We need an easier and more systematic way to compute F^+

Closure of a Set of Attributes (I)

- So far, we have solved the question for A, $B \subseteq R$

  Is $A \rightarrow B \notin F$ logically implied by F?

  by a containment test checking whether $A \rightarrow B \in F^+$ holds

- Problem: Explicit calculation of F^+ is too expensive

- Different, easier, and more efficient solution: Calculation of the closure A^+ of the attribute set A under F
  - A^+ consists of all attributes that are functionally determined by A, i.e., $A \rightarrow A^+$
  - $\forall\ C \subseteq A^+ : A \rightarrow C \in F^+$ (due to decomposition rule)
  - To check if $A \rightarrow B$ is logically implied by F, we check if $B \subseteq A^+$ holds since then $A \rightarrow B \in F^+$ holds