Third Normal Form (3NF) Decomposition (I)

- The algorithm presented is called the 3NF synthesis algorithm

- This algorithm yields a decomposition of a (universal) relation schema *R* with the set *F* of FDs into relation schemas R_1, ..., R_n so that the following three criteria are fulfilled:
  - The decomposition is lossless
  - The decomposition is dependency-preserving
  - Each relation schema R_i ($1 \leq i \leq n$) of the decomposition is in the 3NF

- The proof for correctness can be found in the textbook [SKS]

- The 3NF synthesis algorithm does not check whether *R* is already in the 3NF: Use the algorithm *RelationSchemaIsIn3NF* for this purpose

- Interesting: Frequently, the 3NF synthesis algorithm yields a decomposition into relation schemas R_1, ..., R_n where all R_i are not only in the 3NF but even in the BCNF

Third Normal Form (3NF) Decomposition (II)

- 3NF synthesis algorithm

  $\{(R_1, F_1), \ldots, (R_n, F_n)\}$ *Calculate3NFDecomposition*(R, F)
  // Input: A relation schema R and a set F of FDs on R
  // Output: A decomposition of R into the 3NF relation schemas $R_1, \ldots, R_n$
  // with the corresponding sets $F_1, \ldots, F_n$ of FDs
  // Step 1: Find a minimal cover F_c for F in nonstandard form (i.e., left reduction of the FDs, right reduction of the remaining FDs, removal of FDs of the form $A \rightarrow \emptyset$, union rule for identical left-hand sides)
  F_c := *CalculateMinimalCover*(F)

  // Step 2: Generate relation schemas from the FDs in F_c by forming the union of the attribute sets on both sides of each FD
  $i := 0$
  for each $X \rightarrow Y$ **in** F_c **do**
  $i := i + 1$; $R_i := X \cup Y$
  $F_i := \{A \rightarrow D \mid \exists\, A \rightarrow B \in F_c : A \subseteq R_i \wedge D \subseteq B \wedge D \neq \emptyset \wedge D \cap R_i = D \wedge B \backslash D \cap R_i = \emptyset\}$ // Largest subset D of B must be contained in R_i
  $n := i$ // Number of generated relation schemas

Third Normal Form (3NF) Decomposition (III)

- 3NF synthesis algorithm (*continued*)

  // Step 3: If none of the generated relation schemas contains a candidate key, create an additional relation schema that contains a(n arbitrary) candidate key

  $\{K_1, \ldots, K_m\} := \mathit{CalculateAllCandidateKeys}(R, F_c)$
  found := *false*
  $i := 1$
  while not *found* **and** $i <= m$ **do** // Traverse all candidate keys $K_1, \ldots, K_m$
      $j := 1$
      while not *found* **and** $j <= n$ **do** // Traverse all relation schemas $R_1, \ldots, R_n$
          if $K_i \subseteq R_j$ **then** *found* := *true* **else** $j := j + 1$
      if not *found* **then** $i := i + 1$

  if not *found* **then** // Add an additional schema with the candidate key K_1
      $n := n + 1$
      $R_n := K_1$
      $F_n := \emptyset$

Third Normal Form (3NF) Decomposition (IV)

- 3NF synthesis algorithm (*continued*)

// Step 4: Remove redundant relation schemas, i.e., relation schemas that are contained in another relation schema

i := 1	
while i <= n – 1 **do**	// Traverse all relation schemas R_i, …, $R_{n\text{-}1}$
j := i + 1	// Compare R_i with R_{i+1}
visitIndexIAgain := *false*	// Keep if schema R_i has to be later considered again
while not *visitIndexIAgain* **and** j <= n **do**	// Compare to all relation schemas R_{i+1}, …, R_n
if $R_i \subseteq R_j$ **then**	// Case 1: R_i is contained in R_j
$R_i := R_n$; $F_i := F_n$	// Delete R_i / F_i by replacing it with R_n / F_n
$n := n – 1$	// Reduce the number of relation schemas
visitIndexIAgain := *true*	// The new R_i has still to be checked
else if $R_i \supset R_j$ **then**	// Case 2: R_i contains R_j
if $j < n$ **then** $R_j := R_n$; $F_j := F_n$	// Delete R_j / F_j by replacing it with R_n / F_n
else $R_j := \varnothing$; $F_j := \varnothing$	// or by setting R_j / F_j to the empty set
$n := n – 1$	// Reduce the number of relation schemas
$j := j + 1$	// Continue to compare R_i with R_{j+1}
else	// Case 3: R_i is unequal to R_j
$j := j + 1$	// Continue to compare R_i with R_{j+1}
if not *visitIndexIAgain* **then**	// If R_i has been replaced by R_n, explore index i again
$i := i + 1$	// Otherwise, continue with R_{i+1}

Third Normal Form (3NF) Decomposition (V)

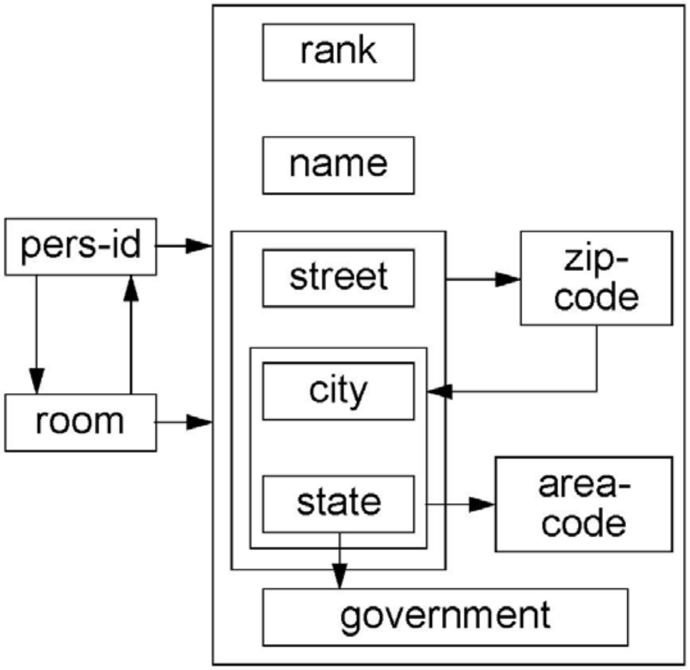
- 3NF synthesis algorithm (*continued*)

  // Step 5: Return the decomposition $R_1, \ldots, R_n$ together with their respective
  // sets $F_1, \ldots, F_n$ of FDs as the result

  return $\{(R_1, F_1), \ldots, (R_n, R_n)\}$

- Example: Universal relation schema *ProfAddr*(pers-id, name, rank, room, city, street, zipcode, area-code, state, government)

- Assumptions
  - A city denotes the residence of a professor.
  - Government is the party of the president.
  - City names are unique within a state.
  - The zipcode does not change within a street.
  - Cities and streets lie completely in the single states.
  - A professor has exactly one office that he does not share
  - …



Third Normal Form (3NF) Decomposition (VI)

- {pers-id} and {room} are candidate keys of the relation *ProfAddr*
- The relation is not in the 3NF since, e.g., the FD {city, state} $\rightarrow$ {area-code} violates the 3NF
- Step 1: Computation of a minimal cover (precomputed)
  - f_1 = {pers-id} $\rightarrow$ {name, rank, room, city, street, state}
  - f_2 = {room} $\rightarrow$ {pers-id}
  - f_3 = {city, street, state} $\rightarrow$ {zipcode}
  - f_4 = {city, state} $\rightarrow$ {area-code}
  - f_5 = {state} $\rightarrow$ {government}
  - f_6 = {zipcode} $\rightarrow$ {city, state}
- Step 2: Generation of relation schemas from the FDs f_1 to f_6
  - From f_1 = {pers-id} $\rightarrow$ {name, rank, room, city, street, state} we obtain:
    - R_1 = {pers-id, name, rank, room, city, street, state}
    - $F_1 = \{f_1, f_2\}$

Third Normal Form (3NF) Decomposition (VII)

- Step 2: Generation of relation schemas from the FDs f_1 to f_6 (*continued*)
  - From f_2 = {room} $\rightarrow$ {pers-id} we obtain:
    - R_2 = {room, pers-id}
    - $F_2 = \{f_2\}$
  - From f_3 = {city, street, state} $\rightarrow$ {zipcode} we obtain:
    - R_3 = {city, street, state, zipcode}
    - $F_3 = \{f_3, f_6\}$
  - From f_4 = {city, state} $\rightarrow$ {area-code} we obtain:
    - R_4 = {city, state, area-code}
    - $F_4 = \{f_4\}$
  - From f_5 = {state} $\rightarrow$ {government} we obtain:
    - R_5 = {state, government}
    - $F_5 = \{f_5\}$

Third Normal Form (3NF) Decomposition (VIII)

- Step 2: Generation of relation schemas from the FDs f_1 to f_6 (*continued*)
  - From f_6 = {zipcode} $\rightarrow$ {city, state} we obtain:
    - R_6 = {zipcode, city, state}
    - $F_6 = \{f_6\}$

- Step 3: Check if a relation schema contains a candidate key
  - The candidate keys are K_1 = {pers-id} and K_2 = {room}
  - Both keys are contained in relation schema R_1

- Step 4: Test for containment relationships between the schemas R_1 to R_6
  - {room, pers-id} $\subseteq$ {pers-id, name, rank, room, city, street, state} [$R_2 \subseteq R_1$]
  - {zipcode, city, state} $\subseteq$ {city, street, state, zipcode} [$R_6 \subseteq R_3$]

Third Normal Form (3NF) Decomposition (IX)

- Step 5: Return the decomposition after renumbering the relation schemas and FDs

$\{(R_1, F_1), (R_2, F_2), (R_3, F_3), (R_4, F_4)\}$ =

{

({pers-id, name, rank, room, city, street, state},
{{pers-id} $\rightarrow$ {name, rank, room, city, street, state},
{room} $\rightarrow$ {pers-id}}),

({city, street, state, zipcode},
{{city, street, state} $\rightarrow$ {zipcode},
{zipcode} $\rightarrow$ {city, state} }),

({city, state, area-code},
{{city, state} $\rightarrow$ {area-code} }),

({state, government},
{{state} $\rightarrow$ {government} })

}

Boyce-Codd Normal Form (BCNF) Decomposition (I)

- A BCNF decomposition of a (universal) relation schema R with the set F of FDs into relation schemas R_1, ..., R_n fulfills the following two criteria:
  - The decomposition is lossless
  - Each relation schema R_i ($1 \leq i \leq n$) of the decomposition is in the BCNF

- This means: We cannot always find a BCNF decomposition which is also dependency-preserving

- But: This case is rare in practice

- The BCNF decomposition algorithm below includes the check if relation schema R and any relation schema R_i of the decomposition is already in the BCNF
  - It uses the predicate *RelationSchemaIsInBCNF* for this purpose
  - Note that this predicate does not work for the decomposed schemas R_i if F is used as a basis (see example below)
  - To work correctly, the restriction F_i of F for R_i has to be taken as a basis

Boyce-Codd Normal Form (BCNF) Decomposition (II)

- Algorithm for computing the restriction of F for a relation schema $S \subseteq R$

F_S *FDRestriction*(F, S)
// Input: A set F of FDs on R and a relation schema $S \subseteq R$
// Output: The restriction F_S of F for S

$F_S := \varnothing$

// Compute the attribute closures of all subsets of S with attributes in S only,
// construct nontrivial FDs from them, and insert them into F_S
for each $X \subseteq S$ **do**

// Compute the attribute closure of X under F_S
$X^+ := \mathit{CalculateAttributeClosure}(F, X) \cap S$ // $X^+ \subseteq S$ holds

// Add a nontrivial FD in F_S for the left-hand side X with respect to X^+
if $X^+ \supset X$ **then** $F_S := F_S \cup \{X \rightarrow (X^+ - X)\}$ // if-condition excludes trivial FDs

// Compute the minimal cover of F_S to minimize the FDs
$F_c := \mathit{CalculateMinimalCover}(F_S)$ // Note that $F_c \equiv F_S$
$F_S := F_c$ // Select F_c for F_S

return F_S