Second Normal Form (2NF) (I)

- Equivalent definitions
  - A relation schema *R* is in the second normal form (2NF) with respect to a set *F* of FDs if, and only if, it is in 1NF and every nonprime attribute *A* in *R* is fully functionally dependent on *every* candidate key of *R*
  - A relation schema *R* is in the second normal form (2NF) with respect to a set *F* of FDs if, and only if, it is in 1NF and every nonprime attribute *A* in *R* is not partially functionally dependent on *any* candidate key of *R*
  - A relation schema *R* is in the second normal form (2NF) with respect to a set *F* of FDs if, and only if, it is in 1NF and for every candidate key *K* of *R* and for every nonprime attribute *A* in *R* the FD $K \to A$ is left-reduced
- Formal definitions of the terms “fully functionally dependent”, “partially functionally dependent”, and “left-reduced” have been provided before
- The 2NF
  - only applies to relation schemas with composite keys, i.e., keys that are composed of two or more attributes
  - holds automatically for relation schemas with only single-attribute keys

Second Normal Form (2NF) (II)

- Example
  - Given: The relation schema *StudentsLecture*(reg-id, id, name, sem)
  - This schema corresponds to the natural join of the relation schemas *attends* and *students*
  - The (primary) key is {reg-id, id} with all FDs having this key on the left-hand side
  - In particular the FDs {reg-id, id} → {name} and {reg-id, id} → {sem} hold

    But additionally the FDs: {reg-id} → {name} and {reg-id} → {sem} hold

    ⇒ violation of the 2NF
  - The following anomalies can occur:
    - *Insertion anomaly*: What do we do with students who do not attend any lecture?
    - *Update anomaly*: If a student reaches the next semester, we must ensure that in all tuples of *StudentsLecture* the semester number is incremented accordingly
    - *Deletion anomaly*: What happens if a student drops her only lecture?

Second Normal Form (2NF) (III)

- Example (*continued*)
  - Solution of these problems:
    - Decompose the relation schema into several relation schemas that all fulfil the 2NF
    - Split *StudentsLecture* in the two schemas *attends*(<u>reg-id</u>, <u>id</u>) and *students*(<u>reg-id</u>, name, sem); both schemas satisfy the 2NF

- Problem of the 2NF that makes it uninteresting in practice
  - It is still possible for a relation schema in the 2NF to exhibit transitive dependencies, i.e., one or more nonprime attributes may be functionally dependent on other nonprime attributes
  - Example
    - Relation schema *LectureProf*(<u>id</u>, title, pers-id, room) with the FD {pers-id} → {room}; both *pers-id* and *room* are nonprime attributes
    - Transitive dependency {id} → {room} exists since {id} → {pers-id} and {pers-id} → {room} hold
  - Therefore, we do not provide a normalization algorithm into the 2NF

Third Normal Form (3NF) (I)

- The 2NF still allows transitive dependencies
- To illustrate why they are problematic, we take up a recent example:
  - Relation schema *LectureProf*(id, title, pers-id, room) with the additional FD {pers-id} → {room}; both *pers-id* and *room* are nonprime attributes
  - Transitive dependency {id} → {room} exists since the FDs {id} → {pers-id} and {pers-id} → {room} hold
  - The following anomalies can occur:
    - *Insertion anomaly*: Information about a professor and his/her room number are not available without the assignment of a lecture
    - *Update anomaly*: A change of the room number of a professor requires a change for each course held by that professor
    - *Deletion anomaly*: If a professor does not hold a class any more, all information about the professor and his/her room number is removed from the database
  - Solution: Splitting of the schema *LectureProf* into the two 3NF schemas *lecture*(id, title, pers-id) and *prof*(pers-id, room)

Third Normal Form (3NF) (II)

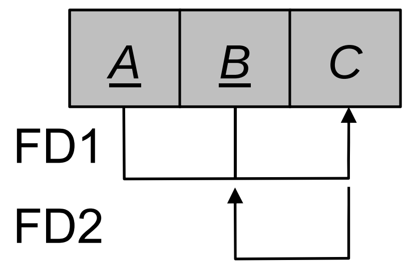
- Conclusion: The goal of the 3NF is to eliminate the dependencies from nonprime attributes
- Equivalent definitions
  - A relation schema R is in the third normal form (3NF) with respect to a set F of FDs if, and only if, it is in 2NF *and* no nonprime attribute A in R is transitively dependent on *any* candidate key of R
  - A relation schema R is in the third normal form (3NF) with respect to a set F of FDs if, and only if, for each FD $X \rightarrow Y$ in F^+ with $X \subseteq R$ and $Y \subseteq R$ at least one of the following conditions holds:
    - $X \rightarrow Y$ is a trivial FD (i.e., $Y \subseteq X$ holds), or
    - X is a superkey of R, or
    - Every element of $Y - X$ is a prime attribute (i.e., contained in some candidate key) of R

Third Normal Form (3NF) (III)

- Equivalent definitions (*continued*)
  - A relation schema R is in the third normal form (3NF) with respect to a set F of FDs if, and only if, for each *left-reduced* FD $X \rightarrow Y$ in F^+ with $X \subseteq R$ and $Y \subseteq R$ at least one of the following conditions holds:
    - $X \rightarrow Y$ is a trivial FD (i.e., $Y \subseteq X$ holds), or
    - X is a *candidate key* of R, or
    - Every element of $Y - X$ is a prime attribute (i.e., contained in some candidate key) of R
- The second and third definition
  - bypass the 2NF and can be applied *directly* to test whether a relation schema is in the 3NF; they do *not* have to go through 2NF first; of course, 1NF is assumed to hold
  - exclude nontrivial FDs between nonprime attributes, i.e., a transitive dependency of the type $A \rightarrow B$ and $B \rightarrow C$, where A is a candidate key, B is not part of or equal to a candidate key (and therefore consists of nonprime attributes only), and C contains at least one nonprime attribute is forbidden

Third Normal Form (3NF) (IV)

- The third condition of the second and third definition
  - does not say that a single candidate key must contain all the attributes in $Y - X$; each attribute in $Y - X$ may be contained in a *different* candidate key
  - is rather unintuitive but ensures that every relation schema has a dependency-preserving decomposition into the 3NF, i.e., the attributes on the left-hand side and right-hand side of each FD can be found in one of the relation schemas of the decomposition (performance issue; later discussed in detail)
  - can be graphically illustrated by the following example (*A* and *B* are prime attributes, *C* is a nonprime attribute):



Third Normal Form (3NF) (V)

- Example
  - Given the schema CarIndex(<u>manufacturer</u>, <u>model-id</u>, manufacturer-id)
  - Consider the FDs:
    - FD1: {model-id, manufacturer} $\rightarrow$ {manufacturer-id}
      [fulfills Condition 2 of 3NF]
    - FD2: {manufacturer-id} $\rightarrow$ {manufacturer}
      [fulfills Condition 3 of 3NF]
  - Relation schema is in 3NF
  - Dependency preservation is ensured since all attributes in FD1 and FD2 are in *CarIndex*

Third Normal Form (3NF) (VI)

- Algorithm to check if a relation schema R with a set F of FDs is in the 3NF

bool RelationSchemaIsIn3NF(R, F)
// Input: A relation schema R and a set F of FDs on R
// Output: *true*, if the relation schema is in the 3NF; *false*, otherwise

$S := \emptyset$ // Stores those FDs that are not trivial and do not have a superkey on their left-hand side
for each $X \to Y$ **in** F **do**
    if not ($Y \subseteq X$) **and not** ($X^+ = R$) **then** // Conditions 1 and 2 of the 3NF are not fulfilled
        $S := S \cup \{X \to Y\}$
if $S = \emptyset$ **then**
    return *true* // No violation detected and no possible Condition 3 case: R is in the 3NF
else
    // Determine all prime attributes
    K := *CalculateAllCandidateKeys*(R, F)
    PrimeAttributes := $\emptyset$
    for each C **in** K **do**
        PrimeAttributes := *PrimeAttributes* $\cup$ C

    // Check the FDs in S with respect to Condition 3 of the 3NF
    for each $X \to Y$ **in** S **do**
        for each A **in** $Y - X$ **do**
            if $A \notin$ *PrimeAttributes* **then return** *false* // Violation of Condition 3: R is not in the 3NF
return *true* // No violation of Condition 3 detected for the FDs in S: R is in the 3NF

Boyce-Codd Normal Form (BCNF) (I)

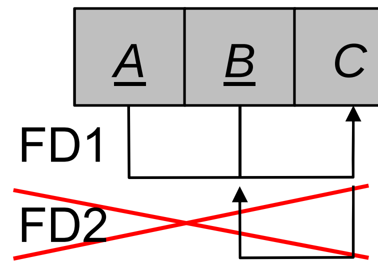
- The 3NF is still not free of anomalies
- To illustrate this, we take up our last schema in 3NF:
  - Given the schema *CarIndex*(<u>model-id</u>, <u>manufacturer</u>, manufacturer-id)
  - Consider the FDs:
    - FD1: {model-id, manufacturer} $\rightarrow$ {manufacturer-id}
    - FD2: {manufacturer-id} $\rightarrow$ {manufacturer}
  - The following anomalies can arise:
    - Insertion of the same manufacturer with different manufacturer ids (and different model ids) is possible
    - 1:1-relationship between *manufacturer* and *manufacturer-id* is connected to model-id
  - Solution: Splitting *CarIndex* into *Producer*(<u>manufacturer-id</u>, manufacturer) and *CarIndexNew*(<u>manufacturer-id</u>, <u>model-id</u>) makes sense since *CarIndex* = *Producer* $\bowtie$ *CarIndexNew*; both schemas are in the BCNF
  - Problem: Split is *not* dependency preserving; FD2 can be checked on relation *Producer* but for checking FD1 a join is needed

Boyce-Codd Normal Form (BCNF) (II)

- The BCNF eliminates all redundancies that can be discovered based on FDs
- Note: There are other types of redundancies not based on FDs that occur very rarely in practice and that we will not consider in this course
- The BCNF is stricter than the 3NF
- Equivalent definitions
  - A relation schema R is in the Boyce-Codd normal form (BCNF) with respect to a set F of FDs if, and only if, for each FD $X \rightarrow Y$ in F^+ with $X \subseteq R$ and $Y \subseteq R$ at least one of the following conditions holds:
    - $X \rightarrow Y$ is a trivial FD (i.e., $Y \subseteq X$ holds), or
    - X is a superkey of R
  - A relation schema R is in the Boyce-Codd normal form (BCNF) with respect to a set F of FDs if, and only if, for each *left-reduced* FD $X \rightarrow Y$ in F^+ with $X \subseteq R$ and $Y \subseteq R$ at least one of the following conditions holds:
    - $X \rightarrow Y$ is a trivial FD (i.e., $Y \subseteq X$ holds), or
    - X is a *candidate key* of R

Boyce-Codd Normal Form (BCNF) (III)

- The third condition of the 3NF has been removed



- To test whether a relation schema is in the BCNF, we determine whether all left-hand sides of FDs are candidate keys
- Example
  - The schemas *Producer*(<u>manufacturer-id</u>, manufacturer) and *CarIndexNew*(<u>manufacturer-id</u>, <u>model-id</u>) as the result of splitting the schema *CarIndex*(<u>manufacturer</u>, manufacturer-id, <u>model-id</u>) are both in the BCNF since *manufacturer-id* is the primary key of *Producer* and *manufacturer* and *model-id* together form the primary key of *CarIndexNew*
  - However, this decomposition is not dependency preserving

Boyce-Codd Normal Form (BCNF) (IV)

- Algorithm to check if a relation schema R with a set F of FDs is in the BCNF

bool RelationSchemaIsInBCNF(R, F, f)
// Input: A relation schema R and a set F of FDs on R
// Output: *true*, if the relation schema is in the BCNF
// *false*, otherwise
// As an output parameter (side effect): $f = X \to Y$ that first violates
// the BCNF
for each $X \to Y$ **in** F **do**
if not ($Y \subseteq X$) **and not** ($X^+ = R$) **then**
// Violation of the two conditions for the BCNF: The FD $X \to Y$ is not
// trivial and does not have a superkey on its left-hand side
$f := X \to Y$
return *false*
return *true* // No violation of the two conditions of the BCNF

Call: InBNCF := *RelationSchemaIsInBCNF*(R, F, f) (InBNCF $\in$ *bool*)

Correctness Criteria for the Normalization Process (I)

- The normalization process eliminates weaknesses (redundancies, inconsistencies, update, insertion and deletion anomalies) of a relation schema R violating a selected normal form by decomposing R into n relation schemas $R_1, ..., R_n$ such that all R_i satisfy the requirements of that normal form

- Important:
  - Normal forms, when considered *in isolation* from other aspects, do not ensure a good database design
  - It is insufficient to check separately that each relation schema in the database is in the 3NF or the BCNF
  - All schemas resulting from a decomposition must be regarded together

Correctness Criteria for the Normalization Process (II)

- Two properties that the resulting relation schemas, taken together, should possess are relevant for the normalization process through decomposition:
  - Lossless (join) decomposition / nonadditive (join) decomposition: Any relation $r(R)$ must be *reconstructable* from the relations $r_1(R_1)$, …, $r_n(R_n)$ of the decomposition and may thus not result in the creation of spurious tuples
  - Dependency preservation: Each FD that holds for the relation schema R must be "represented" in some relation schema $S \in \{R_1, \ldots, R_n\}$ after the decomposition such that the attributes on both sides of the FD are elements of S

- The property of losslessness is mandatory and must be achieved at any rate

- The property of dependency preservation is desirable but sometimes sacrificed