Normal Forms (NFs) and Normalization Algorithms

The Process and Goals of Normalization (I)

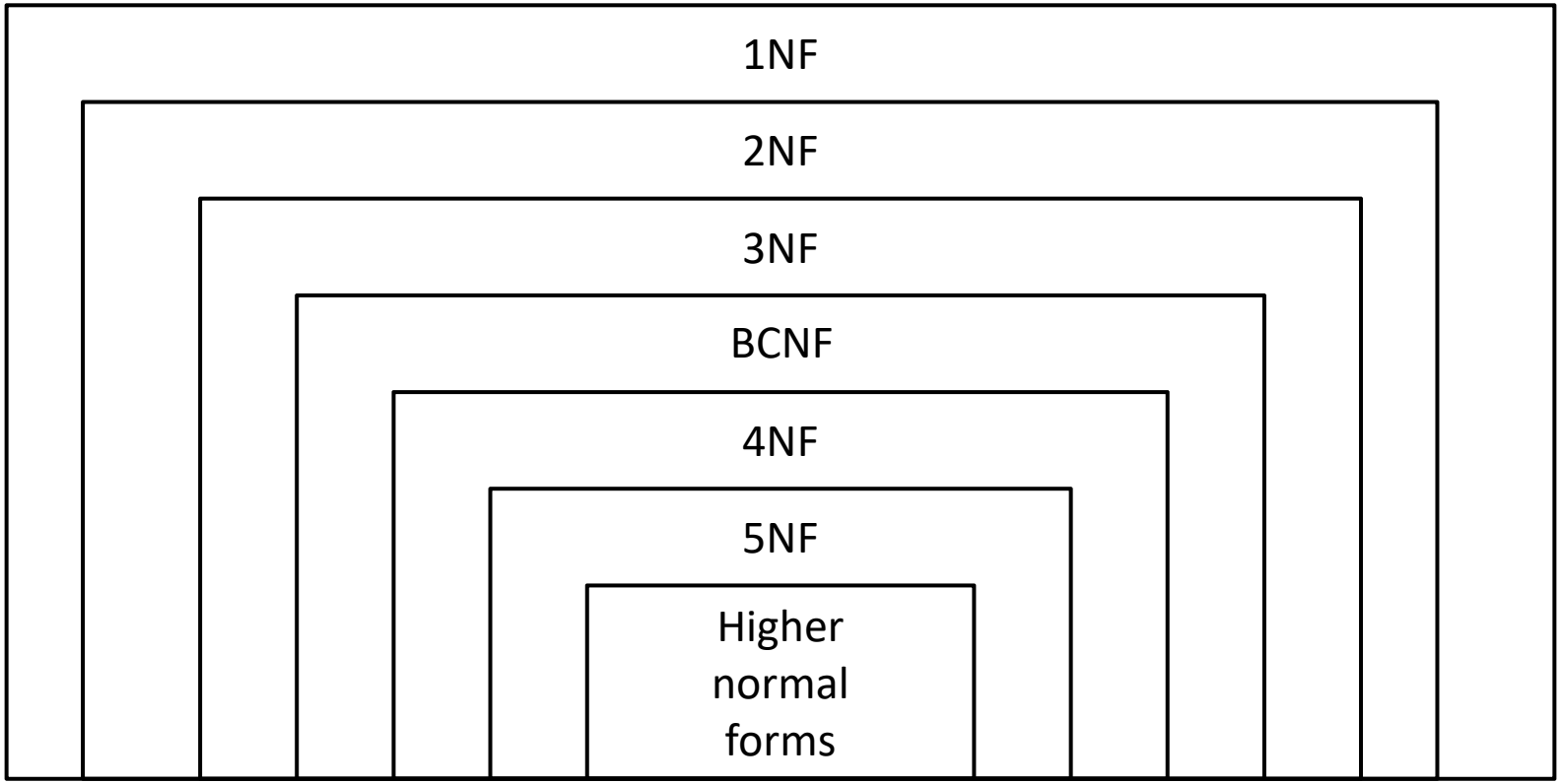
- The normalization process
  - has high practical relevance since it avoids inconsistencies and anomalies of database schemas (= set of relation schemas), which can become problematic for database application programs and user interfaces, to different degrees
  - can be considered as a “filtering” or “purification” process to improve the quality of the database schema design successively
  - proceeds in a top-down fashion by evaluating each relation schema against the requirements for normal forms and by applying a normalization algorithm, if necessary, which decomposes a relation schema into an equivalent set of smaller relation schemas that all satisfy the criteria of a desired normal form
  - is a *formal technique* for analyzing relation schemas based on their primary keys (or candidate keys) and functional dependencies

The Process and Goals of Normalization (II)

- The normalization process (*continued*)
  - involves a set of rules for testing individual relation schemas with respect to desired properties and identifying structural weaknesses
  - *certifies* whether a given relation schema satisfies a certain *normal form*
  - can be considered as relational design by analysis
- The normal form of a relation schema refers to the highest normal form conditions that it meets, and therefore indicates the degree to which it is normalized
- Classification of normal forms considered in this course
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
  - Boyce-Codd normal form (BCNF)
- Each normal form has its specific normalization criteria
- The practical utility of higher normal forms (4NF, 5NF, …) is questionable

The Process and Goals of Normalization (III)

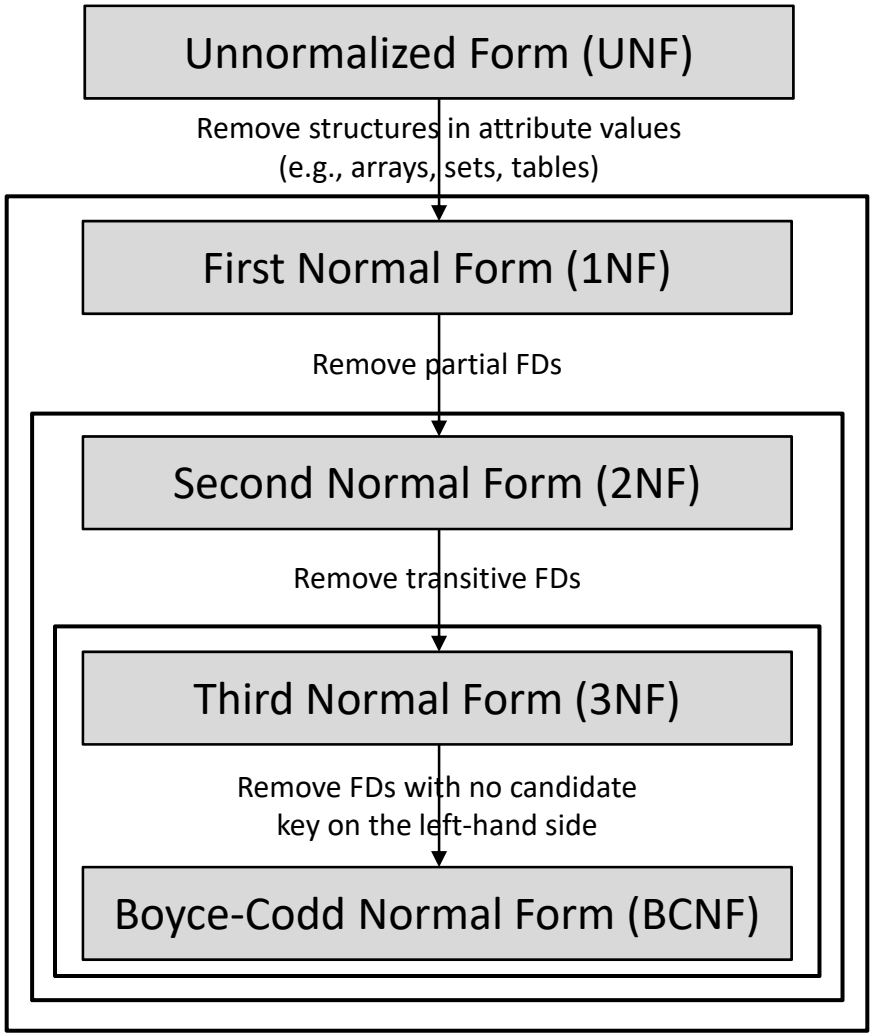
- Diagrammatic illustration of the relationship between the normal forms



- Higher normal forms produce more restricted relation schemas and are less vulnerable to redundancy and update anomalies
- Lower normal forms properly contain higher normal forms

The Process and Goals of Normalization (IV)

- Diagrammatic illustration of the process of normalization



Unnormalized Form (UNF) (I)

- A relation schema is in unnormalized form (UNF) if, and only if, the domain of at least one of its attributes contains non-atomic (or divisible) values
- Examples
  - An attribute value is a *set* of values
  - An attribute value is a composite value (hierarchy)
  - An attribute value is a *list* or *tuple* or *array* of values
  - An attribute value is a table
  - A table contains a nested table (NF^2-relations (NF^2 = Non First Normal Form); nested relations)

parents		
father	mother	children
Ben	Martha	{Liza, Lucia}
Ben	Maria	{Theo, Josef}
John	Martha	{Cleo}

parents			
father	mother	children	
		cname	cage
Ben	Martha	Liza	5
		Lucia	3
Ben	Maria	Theo	3
		Josef	1
John	Martha	Cleo	9

Unnormalized Form (UNF) (II)

- The transformation of an unnormalized table into a flat or unnested table in 1NF depends on the situation
- Example

parents		
father	mother	children
Ben	Martha	{Liza, Lucia}
Ben	Maria	{Theo, Josef}
John	Martha	{Cleo}

parents			
father	mother	children	
		cname	cage
Ben	Martha	Liza	5
		Lucia	3
Ben	Maria	Theo	3
		Josef	1
John	Martha	Cleo	9

Unnesting

parents		
pid	father	mother
354	Ben	Martha
123	Ben	Maria
652	John	Martha

children		
cid	cname	cage
968	Liza	5
385	Lucia	3
712	Theo	3
503	Josef	1
824	Cleo	9

have_kids	
pid	cid
354	968
354	385
123	712
123	503
652	824

First Normal Form (1NF)

- A relation schema is in first normal form (1NF) if, and only if, the domains of all its attributes only contain atomic (or indivisible) values
- In other words: In each relation schema the intersection of each row and each column contains one and only one atomic value
- This fundamental requirement disallows structured and/or complex attribute values as discussed before

StaffBranch	staffNo	sName	position	salary	branchNo	bAddress
	P06	John Smith	Assistant	18000	B04	567 Main St., New York
	P56	Ben Johnson	Manager	30000	B08	122 Glenn Ave., Boston
	P15	Jeff Mayer	Manager	25000	B08	122 Glenn Ave., Boston
	P01	David Ford	Supervisor	27000	B02	333 Ave., Miami
	P04	Mary Lee	Assistant	18000	B08	122 Glenn Ave., Boston
	P87	Ann Brand	Manager	26000	B04	567 Main St., New York

The Notion of Key Revisited

- 2NF, 3NF, and BCNF require a precise understanding of the notion of key
- Definition (again) of different terms around keys
  - A superkey of a relation schema $R(A_1, \ldots, A_n)$ [$R = \{A_1, \ldots, A_n\}$] is a set $S \subseteq R$ such that $S \rightarrow R$
  - A key K is a (minimal) superkey with the additional property: $\nexists L \subset K$ such that $L \rightarrow R$ [alternatively: $\forall\ L \subset K : L \not\rightarrow R$]
  - If a relation schema has more than one key, each of them is called a candidate key
  - One of the candidate keys is *arbitrarily* designated to be the primary key of the relation schema
  - The other candidate keys are called secondary keys
  - An attribute of a relation schema R is called a prime attribute of R if it is a member of *some candidate key* of R
  - An attribute of a relation schema R is called a nonprime attribute of R if it is not a prime attribute, i.e., not a member of any candidate key of R

Determination of All Candidate Keys (I)

- For the definition of the 2NF, 3NF, and BCNF and for the normalization process into the 3NF and the BCNF we have to be able to determine *all candidate keys of a relation schema*

- Example: Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys

- It is obviously difficult to perform this task by "simply looking at the FDs", especially if S contains a large number of FDs

- Conclusion: A systematic consideration is needed for this problem

- The following algorithmic description provides a systematic procedure; it is exponential with the number of attributes

Determination of All Candidate Keys (II)

- Step 1: *Determine the attributes that are neither on the left side nor on the right side of any FD*.
  Reason: These attributes are isolated and cannot be generated by any FD. Therefore, they must belong to any candidate key in order to be able to generate themselves by reflexivity.

- Step 2: *Determine the attributes that are only on the left side of any FD*.
  Reason: These attributes can never be reached by any FD since there is no FD that has them on their right side. Therefore, they must also belong to any candidate key in order to be able to generate themselves by reflexivity.

- Step 3: *Determine the attributes that are only on the right side of any FD*.
  Reason: These attributes can only be reached by other attributes or attribute combinations, i.e., they cannot be part of any candidate key.

- Step 4: *Combine the attributes from steps 1 and 2*.
  Reason: Any candidate key must contain them.

Determination of All Candidate Keys (III)

- Step 5: *Compute the closure of the attributes from step 4. If the attribute closure is equal to all attributes of R, then the attributes from step 4 form the only candidate key, and the algorithm terminates*.
  Reason: No matter how many candidate keys there are, every one of them must contain these attributes, and they already reach all attributes in *R*. Hence they form the only key.

- Step 6: *Otherwise, determine the attributes that are included neither in step 3 nor in step 4*.
  Reason: The attributes from step 3 cannot be part of any candidate key. The attributes from step 4 have already been identified as parts of any candidate key. The remaining attributes are those that appear on *both* sides of an FD.

- Step 7: *Compute the closure of the attributes from step 4 (if there are any) and every combination of attributes from step 6, and determine those minimal attribute closures that are equal to R*.
  Reason: We have to find single missing attributes or the smallest combinations of attributes so that all attributes in *R* can be reached from the resulting attribute sets.

Determination of All Candidate Keys (IV)

- Back to the example: Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys
- Step 1 – Isolated attributes: $\varnothing$
- Step 2 – Attributes only on the left side of any FD: BD
- Step 3 – Attributes only on the right side of any FD: $\varnothing$
- Step 4 – Union of the attributes from step 1 and step 2: BD
- Step 5 – Computation of the closure of the attributes from step 4: $BD^+ = BD$. This does not include all attributes from R, and therefore we have to continue with the next step
- Step 6 – Attributes that can be found on both sides of the FDs: $ACEF$

Determination of All Candidate Keys (V)

Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys

---

- Step 7 – Computation of the closure of the attributes from step 4 and every combination of attributes from step 6
  - We know: The attributes *B* and *D* belong to all candidate keys
  - We construct the power set of the set *ACEF* and map its elements into a list that is increasingly ordered with respect to the number of elements in each set: $\langle A, C, E, F, AC, AE, AF, CE, CF, EF, ACE, ACF, AEF, CEF, ACEF \rangle$
  - We consider the sets with three attributes by adding the sets with one element to *BD*:
    - $BDA^+ = ABD^+ = ABD \subset R$
    - $BDC^+ = BCD^+ = BCDF \subset R$
    - $BDE^+ = ABDE \subset R$
    - $BDF^+ = BCDF \subset R$
  - We have not found any candidate keys by adding one-element sets to *BD*

Determination of All Candidate Keys (VI)

Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys

---

- Step 7 (*continued*)
  - We are processing the list $\langle A, C, E, F, AC, AE, AF, CE, CF, EF, ACE, ACF, AEF, CEF, ACEF \rangle$
  - We consider the sets with four attributes by adding the sets with two elements to BD:
    - $BDAC^+ = ABCD^+ = ABCDEF = R$
    - $BDAE^+ = ABDE^+ = ABDE \subset R$
    - $BDAF^+ = ABDF^+ = ABCDEF = R$
    - $BDCE^+ = BCDE^+ = ABCDEF = R$
    - $BDCF^+ = BCDF^+ = BCDF \subset R$
    - $BDEF^+ = ABCDEF = R$
  - We have found the candidate keys $ABCD$, $ABDF$, $BCDE$, and $BDEF$ by adding two-element sets to BD

Determination of All Candidate Keys (VII)

Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys

---

- Step 7 (continued)
  - Important: We have to continue with those attribute sets (here: *ABDE* and *BCDF*) whose closures are not equal to *R* since there could be a candidate key with five attributes
  - List: $\langle A, C, E, F, AC, AE, AF, CE, CF, EF, ACE, ACF, AEF, CEF, ACEF \rangle$
  - We consider the sets with five attributes by adding the sets with three elements to *BD* (equivalently, we can add one of the two missing attributes to *ABDE* and *BCDF*):
    - $BDACE^+ = ABDEC^+ = ABCDE^+ = ABCD^+ = R$
    - $BDACF^+ = BCDFA^+ = ABCDF^+ = ABCD^+ = R$
    - $BDAEF^+ = ABDEF^+ = ABDF^+ = R$
    - $BDCEF^+ = BCDFE^+ = BCDEF^+ = BCDE^+ = R$
  - We see that we only get superkeys that properly contain candidate keys when we add three-element sets to *BD*.

Determination of All Candidate Keys (VIII)

Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys

---

- Step 7 (continued)
  - When adding *ACEF* to *BD*, we obtain $BDACEF^+ = ABCDEF^+ = R$, which is the *default superkey*; it contains any candidate key found
  - In summary, the candidate keys are: *ABCD*, *ABDF*, *BCDE*, *BDEF*
- Algorithm is time-consuming if it is executed manually
- To accelerate the algorithm, we perform steps 1, 2, 3, 4, and 6 by drawing the following table with the four columns I (isolated attributes), L (attributes on left sides only), B (attributes on both sides), R (attributes on right sides only)

I	L	B	R
–	*B* , *D*	*A* , *C* , *E* , *F*	–

- Perform steps 5 and 7 afterwards

Determination of All Candidate Keys (VIII)

Given the relational schema $R(A, B, C, D, E, F)$ and the set $S = \{DF \rightarrow C, BC \rightarrow F, E \rightarrow A, ABC \rightarrow E\}$ of FDs, determine all candidate keys

---

- We assume that the procedure described above is called by the method

  K := *CalculateAllCandidateKeys*(R, F)

  where R is a relation schema, F is a set of FDs, and K is a set of attribute sets where each set represents a candidate key (in our example, we obtain $K = \{ABCD, ABDF, BCDE, BDEF\} = \{\{A, B, C, D\}, \{A, B, D, F\}, \{B, C, D, E\}, \{B, D, E, F\}\}$)