B0B36DBS: **Database Systems** | Classes 11 and 12: **Functional Dependencies**

01: Closure of a Set of FDs

```
F⁺ = {
  // A1 triviality
  A→A, B→B, C→C,
  AB→A, AB→B, AB→AB, AC→A, AC→C, AC→AC, BC→B, BC→C, BC→BC,
  ABC→A, ABC→B, ABC→C, ABC→AB, ABC→AC, ABC→BC, ABC→ABC,
  // Assumptions
  A→B,
  // A3 composition
  A→AB,
  // A2 transitivity
  AC→B,
  // A3 composition
  AC→AB, AC→BC, AC→ABC
}
```

02: Cover of a Set of FDs

```
F = {
  A→C, // F1
  BC→D, // F2
  C→E, // F3
  E→A // F4
}
G = {
  A→CE, // G1
  C→A, // G2
  E→AE, // G3
  AB→D // G4
}
```

Successful derivation of dependency G1 (A→CE) using all the dependencies in F

```
R1: A→C (F1)
R2: C→E (F3)
R3: A→E (R1, R2, A2 transitivity)
R4: A→CE (R1, R3, A3 composition)
```

Successful derivation of dependency G2 (C→A) using all the dependencies in F

```
R1: C→E (F3)
R2: E→A (F4)
R3: C→A (R1, R2, A2 transitivity)
```

Successful derivation of dependency G3 (E→AE) using all the dependencies in F

```
R1: E→E (A1 triviality)
R2: E→A (F4)
R3: E→AE (R1, R2, A3 composition)
```

Successful derivation of dependency G4 (AB→D) using all the dependencies in F

```
R1: AB→A (A1 triviality)
R2: A→C (F1)
R3: AB→C (R1, R2, A2 transitivity)
R4: AB→B (A1 triviality)
R5: AB→BC (R3, R4, A3 composition)
R6: BC→D (F2)
R7: AB→D (R5, R6, A2 transitivity)
```

Analogously, we also need to verify that every single functional dependency in F can be successfully derived using the dependencies in G

Conclusion: yes, F is a cover of G, as well as G is a cover of F (this relation is symmetrical)

03: Redundant FDs

```
F = {
  AC→B, // F1
  E→B, // F2
  D→C, // F3
  AC→E, // F4
  E→AC // F5
}
```

Successful derivation of dependency F1 (AC→B) using all the remaining dependencies in the original F

```
R1: AC→E (F4)
R2: E→B (F2)
R3: AC→B (R1, R2, A2 transitivity)
```

Successful derivation of dependency F2 (E→B) using all the remaining dependencies in the original F

```
R1: E→AC (F5)
R2: AC→B (F1)
R3: E→B (R1, R2, A2 transitivity)
```

Conclusion: both the dependencies F1 and F2 are redundant when assessed individually, but after one of them is removed, the other will no longer be redundant as a result (F1 was needed for the derivation of F2 and vice versa)

04: Attribute Closures

```
F = {
  AB→D, // F1
  A→CE, // F2
  F→F, // F3
  C→A, // F4
  E→AE // F5
}

A⁺ = {
  A, // A1 triviality
  C, E // F2
}

F⁺ = {
  F // A1 triviality
}

BC⁺ = {
  B, C, // A1 triviality
  A, // F4
  D, // F1
  E // F2
}

ABF⁺ = {
  A, B, F, // A1 triviality
  D, // F1
  C, E // F2
}
```

Observation: ABF is a super-key (since its attribute closure contains all the attributes), but not necessarily a key

05: Cover of a Set of FDs

```
F = {
   A→BEF, // F1
   BC→DE, // F2
   BDE→F, // F3
   ADF→CE, // F4
   E→CBD // F5
}
G = {
   A→B, // G1
   AB→E, // G2
   AD→C, // G3
   BC→E, // G4
   BCE→FD, // G5
   E→C, // G6
   CE→B // G7
}
```

Successful derivation of dependency F1 (A→BEF) using all the dependencies in G

```
A+ = {
   A, // A1 triviality
   B, // G1
   E, // G2
   C, // G6
   F, D // G5
} ⊇ {B, E, F}
```

Analogously for all the remaining functional dependencies in F using G and vice versa

Conclusion: yes, F is a cover of G, as well as G is a cover of F

06: Redundant FDs

```
F = {
   A→C, // F1
   B→A, // F2
   D→AB, // F3
   B→C, // F4
   D→C // F5
}
```

F1 (A→C) is not redundant since A^+ using all the remaining FDs (all except F1) does not contain C

```
A+ using F2, F3, F4 and F5 = {
   A // A1 triviality
}
```

F2 (B→A) is not redundant since B^+ using all the remaining FDs (all except F2) does not contain A

```
B+ using F1, F3, F4 and F5 = {
   B, // A1 triviality
   C // F4
}
```

F3 (D→AB) is not redundant since D^+ using all the remaining FDs (all except F3) does not contain both A and B

```
D+ using F1, F2, F4 and F5 = {
   D, // A1 triviality
   C // F5
}
```

F4 (B→C) is redundant since B^+ using all the remaining FDs (all except F4) contains C, and so F4 can be removed

```
B+ using F1, F2, F3 and F5 = {
   B, // A1 triviality
   A, // F2
   C // F1
} ⊇ {C}
```

F5 (D→C) is also redundant since D^+ using all the remaining FDs (all except F5 and F4) contains C

```
D+ using F1, F2 and F3 = {
   D, // A1 triviality
   A, B, // F3
   C // F1
} ⊇ {C}
```

Conclusion: both F4 (B→C) and F5 (D→C) were redundant and could be removed

07: Redundant Attributes

```
F = {
  AB→D, // F1
  A→CE, // F2
  C→A, // F3
  E→AE, // F4
  F→B, // F5
  BCEF→A // F6
}
```

Attribute A is not redundant in F1 (AB→D) since attribute closure of all the remaining attributes (i.e. just B) does not contain D, and so it cannot be removed

```
B+ = {
  B // A1 triviality
}
```

Attribute B is not redundant in F1 (AB→D), and so it cannot be removed as well

```
A+ = {
  A, // A1 triviality
  C, E // F2
}
```

Conclusion: there are no redundant attributes in F1 (AB→D)

Attribute B is redundant in F6 (BCEF→A), and so F6 can be replaced with F6' (CEF→A)

```
CEF+ = {
  C, E, F, // A1 triviality
  A, // F3
  B, // F5
  D // F1
} ⊇ {A}
```

Attribute C is redundant in F6' (CEF→A), and so F6' can be replaced with F6'' (EF→A)

```
EF+ = {
  E, F, // A1 triviality
  A, // F4
  C, // F2
  B, // F5
  D // F1
} ⊇ {A}
```

Attribute E is not redundant in F6'' (EF→A), and so it cannot be removed

```
F+ = {
  F, // A1 triviality
  B // F5
}
```

Attribute F is redundant in F6'' (EF→A), and so F6'' can be replaced with F6''' (E→A)

```
E+ = {
  E, // A1 triviality
  A, // F4
  C // F2
} ⊇ {A}
```

Conclusion: attributes B, C and F were redundant in F6 (BCEF→A), and so F6 could be replaced with F6''' (E→A)

08: Minimal Cover of a Set of FDs

Solution 1

BC→D, BC→E, DE→B, CE→A, CE→B

Solution 2

BC→A, BC→D, BC→E, DE→B, CE→B

09: Minimal Cover of a Set of FDs

AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CE→G

10: Minimal Cover of a Set of FDs

Solution: there are no redundant attributes and nor redundant dependencies

AB→H, EB→C, BC→A, C→F, F→G, A→E, A→C, E→D

11: First Key

We start with a trivial super-key ABCDE (i.e. a super-key containing all the attributes) and remove all redundant attributes from a trivial functional dependency ABCDE→ABCDE

Attribute A is not redundant in ABCDE→ABCDE

```
BCDE+ = {
   B, C, D, E // A1 triviality
}
```

Attribute B is redundant in ABCDE→ABCDE, and so we obtain a simplified dependency ACDE→ABCDE

```
ACDE+ = {
   A, C, D, E, // A1 triviality
   B // F2 or F3
}
```

Attribute C is not redundant in ACDE→ABCDE

```
ADE+ = {
   A, D, E, // A1 triviality
   B // F2
}
```

Attribute D is redundant in ACDE→ABCDE, and so we obtain a simplified dependency ACE→ABCDE

```
ACE+ = {
   A, C, E, // A1 triviality
   B, // F3
   D // F1
}
```

Attribute E is not redundant in ACE→ABCDE

```
AC+ = {
   A, C // A1 triviality
}
```

Conclusion: the first key is ACE

12: All Keys

Assumption: we already have one key, in particular the first key ACE (see above)

The initial working set of found keys is {ACE}

Step 1: processing of a key ACE (the first not yet processed key from the current working set of keys):

Dependency F1: $BC \rightarrow DE$
Let us have a look at the intersection of the current key with the right side of this dependency
$ACE \cap DE \neq \emptyset$
Since this intersection is not empty, we will find a new key candidate
We take the current key, remove attributes from the right side and add attributes from the left side
$(ACE \setminus DE) \cup BC = AC \cup BC = ABC$
The current working set does not contain even a single key that would be a subset of this candidate
Therefore we continue and remove redundant attributes from ABC in order to obtain a new key
There are no such redundant attributes
Hence ABC is a newly found key, we add it into the current working set of keys

Dependency F2: $DE \rightarrow B$
$ACE \cap B = \emptyset$ and thus this functional dependency cannot be used to find a new key

Dependency F3: $CE \rightarrow B$
$ACE \cap B = \emptyset$

The current working set of found keys is {ACE, ABC}

Step 2: processing of a key ABC:

Dependency F1: $BC \rightarrow DE$
$ABC \cap DE = \emptyset$

Dependency F2: $DE \rightarrow B$
$ABC \cap B \neq \emptyset$
$(ABC \setminus B) \cup DE = AC \cup DE = ACDE$
$ACE \subseteq ACDE$ and therefore this candidate will not be further considered

Dependency F3: $CE \rightarrow B$
$ABC \cap B \neq \emptyset$
$(ABC \setminus B) \cup CE = AC \cup CE = ACE$
$ACE \subseteq ACE$ and therefore this candidate will not be further considered as well

All keys from the working set were successfully processed

Conclusion: {ACE, ABC} are all keys

13: All Keys

ADF, ABF, ACF

14: Normal Forms

The provided relational schema is in 3NF

	1NF	2NF	3NF	BCNF	
$BC \rightarrow D$:	yes	yes	yes	yes	**BCNF**
$BC \rightarrow E$:	yes	yes	yes	yes	**BCNF**
$DE \rightarrow B$:	yes	yes	yes	no	**3NF**
$CE \rightarrow B$:	yes	yes	yes	yes	**BCNF**