0.30 Candidate Keys for Class

you-said

r1 (OfficeLocation, ManagerID, ProjectID, DepartmentID, EmployeeID) FD = { {EmployeeID, DepartmentID, ProjectID} => {ManagerID} {EmployeeID, DepartmentID, ProjectID} => {ManagerID, OfficeLocation} {ManagerID} => {EmployeeID, DepartmentID} {OfficeLocation} => {EmployeeID, ProjectID} } apply this algorithm to the above set of functional dependencies: functional_dependencies (FD) of type -> for ith functional dependency. left = Union of all in FD right = Union of all in FD Initialize case_1, case_2, case_3, case_4 as empty sets. For x in attributes: if x left & x right: case_1.add(x) if x left & x right: case_2.add(x) if x left & x right: case_3.add(x) if x left & x right: case_4.add(x) return attributes in case_1, case_2, case_3, case_4

ChatGPT said:

Let's walk through the algorithm for the given set of functional dependencies (FD):

step-1-extract-the-left-and-right-sides-of-each-functional-dependency.

Step 1: Extract the left and right sides of each functional dependency.

FDs:

1. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}${EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}${EmployeeID,Departm
3. $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}${ManagerID}→{EmployeeID,DepartmentID}
4. $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectID}

step-2-calculate-the-left-and-right-sides.

Step 2: Calculate the left and right sides.

- **Left:** Union of all attributes on the left side of FDs:
    - $\{EmployeeID, DepartmentID, ProjectID, ManagerID, OfficeLocation\}${EmployeeID,Departm
- **Right:** Union of all attributes on the right side of FDs:

  - $\{ManagerID, OfficeLocation, EmployeeID, DepartmentID, ProjectID\}$\{ManagerID,OfficeLoc

So:

- Left: $\{EmployeeID, DepartmentID, ProjectID, ManagerID, OfficeLocation\}$\{EmployeeID,Departm
- Right: $\{ManagerID, OfficeLocation, EmployeeID, DepartmentID, ProjectID\}$\{ManagerID,OfficeLo

step-3-apply-the-conditions-for-cases.

Step 3: Apply the conditions for cases.

- **Case 1:** $x \notin$ left and $x \notin$ rightx/left and x/right

  - No attribute satisfies this condition, as all attributes are either on the left or the right.

- **Case 2:** $x \notin$ left and $x \in$ rightx/left and xright

  - No attribute satisfies this condition, because all right-side attributes are also present in the left side.

- **Case 3:** $x \in$ left and $x \notin$ rightxleft and x/right

  - No attribute satisfies this condition, because all left-side attributes are present in the right side as well.

- **Case 4:** $x \in$ left and $x \in$ rightxleft and xright

  - All attributes are in both the left and right sides: $\{EmployeeID, DepartmentID, ProjectID, Man$

final-output

Final Output:

- Case 1: Empty set
- Case 2: Empty set
- Case 3: Empty set
- Case 4: $\{EmployeeID, DepartmentID, ProjectID, ManagerID, OfficeLocation\}$ {EmployeeID,Depar

So, all the attributes are in **Case 4**.
Would you like to explore this further or need clarification on any part of this?
you-said-1

use this algorithm completely for closure: Closure of under F (+) result := Do result_previous = result For each → in F: If (result): result := result Union While (result_previous <> result) Return result give closures of {EmployeeID}, {DepartmentID}, {ProjectID}, {ManagerID}, {OfficeLocation}
ChatGPT said:
Let's compute the closure for each attribute ({EmployeeID}, {DepartmentID}, {ProjectID}, {ManagerID}, {OfficeLocation}) using the closure algorithm:
given-functional-dependencies-fd

Given Functional Dependencies (FD):

1. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}$ {EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}$ {EmployeeID,Departm
3. $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$ {ManagerID}→{EmployeeID,DepartmentID}
4. $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}$ {OfficeLocation}→{EmployeeID,ProjectID}

closure-algorithm

Closure Algorithm:

- **Step 1:** Initialize `result` to the given attribute(s).
- **Step 2:** Iteratively check each functional dependency. If the left side of the FD is a subset of `result`, add the right side to `result`.
- **Step 3:** Repeat until no new attributes are added to `result`.

---

closure-of-employeeidemployeeid

Closure of $\{EmployeeID\}^{+}${EmployeeID}+:Closure of \{ EmployeeID\}^{+}{EmployeeID}+:

1. **Initial result:** $\{EmployeeID\}${EmployeeID}
2. No functional dependency has $\{EmployeeID\}${EmployeeID} as the complete left-hand side, so no new attributes are added.

- **Final result:** $\{EmployeeID\}${EmployeeID}

---

closure-of-departmentiddepartmentid

Closure of $\{DepartmentID\}^{+}${DepartmentID}+:Closure of \{ DepartmentID\}^{+}{Departmen

1. **Initial result:** $\{DepartmentID\}${DepartmentID}
2. No functional dependency has $\{DepartmentID\}${DepartmentID} as the complete left-hand side, so no new attributes are added.

- **Final result:** $\{DepartmentID\}${DepartmentID}

---

closure-of-projectidprojectid

Closure of $\{ProjectID\}^+${ProjectID}+:Closure of \{ ProjectID\}ˆ{+}{ProjectID}+:

1. **Initial result:** $\{ProjectID\}${ProjectID}
2. No functional dependency has $\{ProjectID\}${ProjectID} as the complete left-hand side, so no new attributes are added.

- **Final result:** $\{ProjectID\}${ProjectID}

---

closure-of-manageridmanagerid

Closure of $\{ManagerID\}^+${ManagerID}+:Closure of \{ ManagerID\}ˆ{+}{ManagerID}+:

1. **Initial result:** $\{ManagerID\}${ManagerID}
2. Apply FD: $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}${ManagerID}→{EmployeeID,Departmen since $\{ManagerID\}${ManagerID} is the left-hand side:
    - Result = $\{ManagerID, EmployeeID, DepartmentID\}${ManagerID,EmployeeID,DepartmentID}
3. No further functional dependencies apply to this set.

- **Final result:** $\{ManagerID, EmployeeID, DepartmentID\}${ManagerID,EmployeeID,DepartmentID}

---

closure-of-officelocationofficelocation

Closure of $\{OfficeLocation\}^+${OfficeLocation}+:Closure of \{ OfficeLocation\}^{+}{OfficeLocati

1. **Initial result:** $\{OfficeLocation\}${OfficeLocation}
2. Apply FD: $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectI since $\{OfficeLocation\}${OfficeLocation} is the left-hand side:

    - Result $= \{OfficeLocation, EmployeeID, ProjectID\}${OfficeLocation,EmployeeID,ProjectID}

3. Now, check if any FD applies to the new result. None of the other FDs apply to this set.

- **Final result:** $\{OfficeLocation, EmployeeID, ProjectID\}${OfficeLocation,EmployeeID,ProjectID}

---

summary-of-closures

Summary of Closures:

- $\{EmployeeID\}^+ = \{EmployeeID\}${EmployeeID}+={EmployeeID}
- $\{DepartmentID\}^+ = \{DepartmentID\}${DepartmentID}+={DepartmentID}
- $\{ProjectID\}^+ = \{ProjectID\}${ProjectID}+={ProjectID}
- $\{ManagerID\}^+ = \{ManagerID, EmployeeID, DepartmentID\}${ManagerID}+={ManagerID,Employ
- $\{OfficeLocation\}^+ = \{OfficeLocation, EmployeeID, ProjectID\}${OfficeLocation}+={OfficeLocatio

This shows the set of attributes functionally implied by each initial attribute. Let me know if you'd like further details or explanations!

you-said-2

use this algorithm completely for closure: Closure of under F (+) result := Do result_previous = result For each → in F: If (result): result := result Union While (result_previous <> result) Return result give closures of {EmployeeID, DepartmentID}, {EmployeeID, ProjectID}, {EmployeeID, ManagerID}, {EmployeeID, OfficeLocation}

ChatGPT said:

Let's compute the closure for the following sets of attributes using the closure algorithm:

- $\{EmployeeID, DepartmentID\}${EmployeeID,DepartmentID}
- $\{EmployeeID, ProjectID\}${EmployeeID,ProjectID}
- $\{EmployeeID, ManagerID\}${EmployeeID,ManagerID}
- $\{EmployeeID, OfficeLocation\}${EmployeeID,OfficeLocation}

given-functional-dependencies-fd-1

Given Functional Dependencies (FD):

1. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}${EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}${EmployeeID,Departm
3. $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}${ManagerID}→{EmployeeID,DepartmentID}
4. $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectID}

---

closure-of-employeeiddepartmentidemployeeiddepartmentid

1. Closure of $\{EmployeeID, DepartmentID\}^+${EmployeeID,DepartmentID}+:1. Closure of \{ EmployeeID,DepartmentID\}^{+}{EmployeeID,DepartmentID}+:

1. **Initial result:** $\{EmployeeID, DepartmentID\}${EmployeeID,DepartmentID}
2. No functional dependency has $\{EmployeeID, DepartmentID\}${EmployeeID,DepartmentID} as the complete left-hand side. The only dependencies that match need an additional $\{ProjectID\}${ProjectID}.
3. No new attributes are added.

- **Final result:** $\{EmployeeID, DepartmentID\}$\{EmployeeID,DepartmentID\}

---

closure-of-employeeidprojectidemployeeidprojectid

2. Closure of $\{EmployeeID, ProjectID\}^{+}$**{EmployeeID,ProjectID}+:2. Closure of \\{ EmployeeID,ProjectID\\}ˆ{+}{EmployeeID,ProjectID}+:**

1. **Initial result:** $\{EmployeeID, ProjectID\}$\{EmployeeID,ProjectID\}
2. No functional dependency has $\{EmployeeID, ProjectID\}$\{EmployeeID,ProjectID\} as the complete left-hand side. The only dependencies that match need an additional $\{DepartmentID\}$\{DepartmentID\}.
3. No new attributes are added.

- **Final result:** $\{EmployeeID, ProjectID\}$\{EmployeeID,ProjectID\}

---

closure-of-employeeidmanageridemployeeidmanagerid

3. Closure of $\{EmployeeID, ManagerID\}^{+}$**{EmployeeID,ManagerID}+:3. Closure of \\{ EmployeeID,ManagerID\\}ˆ{+}{EmployeeID,ManagerID}+:**

1. **Initial result:** $\{EmployeeID, ManagerID\}$\{EmployeeID,ManagerID\}
2. Apply FD: $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$\{ManagerID\}→\{EmployeeID,Departmen since $\{ManagerID\}$\{ManagerID\} is a subset of the current result:
    - Result = $\{EmployeeID, ManagerID, DepartmentID\}$\{EmployeeID,ManagerID,DepartmentID\}
3. Now check if any FD applies to the new set $\{EmployeeID, ManagerID, DepartmentID\}$\{EmployeeID,M None of the remaining dependencies apply because we need $\{ProjectID\}$\{ProjectID\} as part of the left-hand side.

- **Final result:** $\{EmployeeID, ManagerID, DepartmentID\}${EmployeeID,ManagerID,DepartmentID}

---

closure-of-employeeidofficelocationemployeeidofficelocation

4. Closure of $\{EmployeeID, OfficeLocation\}^+${EmployeeID,OfficeLocation}+:4. Closure of \{ EmployeeID,OfficeLocation\}ˆ{+}{EmployeeID,OfficeLocation}+:

1. **Initial result:** $\{EmployeeID, OfficeLocation\}${EmployeeID,OfficeLocation}
2. Apply FD: $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectI
   since $\{OfficeLocation\}${OfficeLocation} is a subset of the current result:
    - Result = $\{EmployeeID, OfficeLocation, ProjectID\}${EmployeeID,OfficeLocation,ProjectID}
3. Now check if any FD applies to the new set $\{EmployeeID, OfficeLocation, ProjectID\}${EmployeeID,Of
   None of the other dependencies apply because we need $\{DepartmentID\}${DepartmentID} as part of the left-hand side.

- **Final result:** $\{EmployeeID, OfficeLocation, ProjectID\}${EmployeeID,OfficeLocation,ProjectID}

---

summary-of-closures-1

Summary of Closures:

- $\{EmployeeID, DepartmentID\}^+ = \{EmployeeID, DepartmentID\}${EmployeeID,DepartmentID}+={
- $\{EmployeeID, ProjectID\}^+ = \{EmployeeID, ProjectID\}${EmployeeID,ProjectID}+={EmployeeID,P
- $\{EmployeeID, ManagerID\}^+ = \{EmployeeID, ManagerID, DepartmentID\}${EmployeeID,ManagerI
- $\{EmployeeID, OfficeLocation\}^+ = \{EmployeeID, OfficeLocation, ProjectID\}${EmployeeID,OfficeL

This shows the set of attributes functionally implied by each initial attribute set. Let me know if you need further clarification or have more examples you'd like to explore!

you-said-3

use this algorithm completely for closure: Closure of under F (+) result := Do result_previous = result For each → in F: If (result): result := result Union While (result_previous <> result) Return result give closures of {DepartmentID, ProjectID}, {DepartmentID, ManagerID}, {DepartmentID, OfficeLocation} , {ProjectID, ManagerID}, {ProjectID, OfficeLocation}, {ManagerID, OfficeLocation}

ChatGPT said:

Let's compute the closure for the following sets of attributes using the closure algorithm:

- $\{DepartmentID, ProjectID\}${DepartmentID,ProjectID}
- $\{DepartmentID, ManagerID\}${DepartmentID,ManagerID}
- $\{DepartmentID, OfficeLocation\}${DepartmentID,OfficeLocation}
- $\{ProjectID, ManagerID\}${ProjectID,ManagerID}
- $\{ProjectID, OfficeLocation\}${ProjectID,OfficeLocation}
- $\{ManagerID, OfficeLocation\}${ManagerID,OfficeLocation}

given-functional-dependencies-fd-2

Given Functional Dependencies (FD):

1. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}${EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}${EmployeeID,Departm
3. $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}${ManagerID}→{EmployeeID,DepartmentID}
4. $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectID}

---

closure-of-departmentidprojectiddepartmentidprojectid

1. Closure of $\{DepartmentID, ProjectID\}^+${DepartmentID,ProjectID}+:1. Closure of \{ DepartmentID,ProjectID\}^{+}{DepartmentID,ProjectID}+:

1. **Initial result:** $\{DepartmentID, ProjectID\}${DepartmentID,ProjectID}

2. No functional dependency has $\{DepartmentID, ProjectID\}${DepartmentID,ProjectID} as the left-hand side. The dependencies need $\{EmployeeID\}${EmployeeID} as part of the left-hand side.
3. No new attributes are added.

- **Final result:** $\{DepartmentID, ProjectID\}${DepartmentID,ProjectID}

---

closure-of-departmentidmanageriddepartmentidmanagerid

2. Closure of $\{DepartmentID, ManagerID\}^{+}${DepartmentID,ManagerID}+:2. Closure of \{ DepartmentID,ManagerID\}^{+}{DepartmentID,ManagerID}+:

1. **Initial result:** $\{DepartmentID, ManagerID\}${DepartmentID,ManagerID}
2. Apply FD: $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}${ManagerID}→{EmployeeID,Departmen since $\{ManagerID\}${ManagerID} is a subset of the current result:
    - Result = $\{DepartmentID, ManagerID, EmployeeID\}${DepartmentID,ManagerID,EmployeeID}
3. No further functional dependencies apply to this result.

- **Final result:** $\{DepartmentID, ManagerID, EmployeeID\}${DepartmentID,ManagerID,EmployeeID}

---

closure-of-departmentidofficelocationdepartmentidofficelocation

3. Closure of $\{DepartmentID, OfficeLocation\}^+$ {DepartmentID,OfficeLocation}+:3. Closure of \{ DepartmentID,OfficeLocation\}ˆ{+}{DepartmentID,OfficeLocation}+:

1. **Initial result:** $\{DepartmentID, OfficeLocation\}$ {DepartmentID,OfficeLocation}
2. Apply FD: $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}$ {OfficeLocation}→{EmployeeID,ProjectI
   since $\{OfficeLocation\}$ {OfficeLocation} is a subset of the current result:

    - Result = $\{DepartmentID, OfficeLocation, EmployeeID, ProjectID\}$ {DepartmentID,OfficeLoca

3. Now check if any FD applies to $\{DepartmentID, OfficeLocation, EmployeeID, ProjectID\}$ {Departme
   Apply FD: $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}$ {EmployeeI
   adding $\{ManagerID\}$ {ManagerID}.

    - Result = $\{DepartmentID, OfficeLocation, EmployeeID, ProjectID, ManagerID\}$ {Department

4. No further functional dependencies apply.

- **Final result:** $\{DepartmentID, OfficeLocation, EmployeeID, ProjectID, ManagerID\}$ {Department

---

closure-of-projectidmanageridprojectidmanagerid

4. Closure of $\{ProjectID, ManagerID\}^+$ {ProjectID,ManagerID}+:4. Closure of \{ ProjectID,ManagerID\}ˆ{+}{ProjectID,ManagerID}+:

1. **Initial result:** $\{ProjectID, ManagerID\}$ {ProjectID,ManagerID}
2. Apply FD: $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$ {ManagerID}→{EmployeeID,Departmen
   since $\{ManagerID\}$ {ManagerID} is a subset of the current result:

    - Result = $\{ProjectID, ManagerID, EmployeeID, DepartmentID\}$ {ProjectID,ManagerID,Emplo

3. No further functional dependencies apply.

- **Final result:** $\{ProjectID, ManagerID, EmployeeID, DepartmentID\}$\{ProjectID,ManagerID,Employ

---

closure-of-projectidofficelocationprojectidofficelocation

5. Closure of $\{ProjectID, OfficeLocation\}^{+}$**\{ProjectID,OfficeLocation\}+:5. Closure of \\{ ProjectID,OfficeLocation\\}ˆ{+}\{ProjectID,OfficeLocation\}+:**

1. **Initial result:** $\{ProjectID, OfficeLocation\}$\{ProjectID,OfficeLocation\}
2. Apply FD: $\{OfficeLocation\} \to \{EmployeeID, ProjectID\}$\{OfficeLocation\}→\{EmployeeID,ProjectII since $\{OfficeLocation\}$\{OfficeLocation\} is part of the current result. But $\{ProjectID\}$\{ProjectID\} is already included in the result, so no new attributes are added.
3. No further functional dependencies apply.

- **Final result:** $\{ProjectID, OfficeLocation, EmployeeID\}$\{ProjectID,OfficeLocation,EmployeeID\}

---

closure-of-manageridofficelocationmanageridofficelocation

6. Closure of $\{ManagerID, OfficeLocation\}^{+}$**\{ManagerID,OfficeLocation\}+:6. Closure of \\{ ManagerID,OfficeLocation\\}ˆ{+}\{ManagerID,OfficeLocation\}+:**

1. **Initial result:** $\{ManagerID, OfficeLocation\}$\{ManagerID,OfficeLocation\}
2. Apply FD: $\{ManagerID\} \to \{EmployeeID, DepartmentID\}$\{ManagerID\}→\{EmployeeID,Departmen since $\{ManagerID\}$\{ManagerID\} is part of the current result:

    - Result $= \{ManagerID, OfficeLocation, EmployeeID, DepartmentID\}$\{ManagerID,OfficeLocati

3. Apply FD: $\{OfficeLocation\} \to \{EmployeeID, ProjectID\}$\{OfficeLocation\}→\{EmployeeID,ProjectII since $\{OfficeLocation\}$\{OfficeLocation\} is part of the result:

  - Result $= \{ManagerID, OfficeLocation, EmployeeID, DepartmentID, ProjectID\}$\{ManagerID,

4. No further functional dependencies apply.

- **Final result:** $\{ManagerID, OfficeLocation, EmployeeID, DepartmentID, ProjectID\}$\{ManagerID,

---

summary-of-closures-2

Summary of Closures:

- $\{DepartmentID, ProjectID\}^{+} = \{DepartmentID, ProjectID\}$\{DepartmentID,ProjectID\}+=\{Depart
- $\{DepartmentID, ManagerID\}^{+} = \{DepartmentID, ManagerID, EmployeeID\}$\{DepartmentID,Man
- $\{DepartmentID, OfficeLocation\}^{+} = \{DepartmentID, OfficeLocation, EmployeeID, ProjectID, M$
- $\{ProjectID, ManagerID\}^{+} = \{ProjectID, ManagerID, EmployeeID, DepartmentID\}$\{ProjectID,M
- $\{ProjectID, OfficeLocation\}^{+} = \{ProjectID, OfficeLocation, EmployeeID\}$\{ProjectID,OfficeLocat
- $\{ManagerID, OfficeLocation\}^{+} = \{ManagerID, OfficeLocation, EmployeeID, DepartmentID, Pr$

Let me know if you'd like further clarification!
you-said-4

use this algorithm completely for closure: Closure of under F (+) result := Do result_previous = result For each → in F: If (result): result := result Union While (result_previous <> result) Return result give closure of {ProjectID,ManagerID}

ChatGPT said:

Let's compute the closure of $\{ProjectID, ManagerID\}^{+}$\{ProjectID,ManagerID\}+ using the closure algorithm step by step.

given-functional-dependencies-fd-3

Given Functional Dependencies (FD):

1. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}$ {EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}$ {EmployeeID,Departm
3. $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$ {ManagerID}→{EmployeeID,DepartmentID}
4. $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}$ {OfficeLocation}→{EmployeeID,ProjectID}

---

steps-for-closure-of-projectidmanageridprojectidmanagerid

Steps for Closure of $\{ProjectID, ManagerID\}^{+}$ {ProjectID,ManagerID}+:Steps for Closure of \{ ProjectID,ManagerID\}ˆ{+}{ProjectID,ManagerID}+:

1. **Initial result:** $\{ProjectID, ManagerID\}$ {ProjectID,ManagerID}
2. **Step 1:** Apply FD $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$ {ManagerID}→{EmployeeID,De
    - Since $\{ManagerID\}$ {ManagerID} is a subset of the current result $\{ProjectID, ManagerID\}$ {Proj we can apply this functional dependency and add $\{EmployeeID, DepartmentID\}$ {EmployeeID,De
    - Result = $\{ProjectID, ManagerID, EmployeeID, DepartmentID\}$ {ProjectID,ManagerID,Employ
3. **Step 2:** Check remaining FDs:
    - FD $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}$ {EmployeeID,DepartmentID,P can be applied because $\{EmployeeID, DepartmentID, ProjectID\}$ {EmployeeID,DepartmentID,P is now a subset of the current result.
        - This FD adds $\{ManagerID\}$ {ManagerID}, but $\{ManagerID\}$ {ManagerID} is already present in the result, so no new attributes are added.

- FD $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}${EmployeeID can also be applied since $\{EmployeeID, DepartmentID, ProjectID\}${EmployeeID,DepartmentID, is a subset of the current result.

  - This FD adds $\{OfficeLocation\}${OfficeLocation}.
  - Result $= \{ProjectID, ManagerID, EmployeeID, DepartmentID, OfficeLocation\}${Projec

4. **Step 3:** Apply FD $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,P

   - Since $\{OfficeLocation\}${OfficeLocation} is now a subset of the current result, we apply this FD, but $\{EmployeeID, ProjectID\}${EmployeeID,ProjectID} are already present in the result.
   - No new attributes are added.

5. **Termination:** Since no new attributes were added in the last step, the closure is complete.

---

final-result

Final Result:

- **Closure of** $\{ProjectID, ManagerID\}^+$**{ProjectID,ManagerID}+** $= \{ProjectID, ManagerID, En$

This shows all the attributes implied by $\{ProjectID, ManagerID\}${ProjectID,ManagerID} based on the given functional dependencies.

you-said-5

So the sets of 2 attributes whose closure contains all attributes are {DepartmentID,OfficeLocation} , {ProjectID,ManagerID} and {ManagerID,OfficeLocation}. To find minimal candidate keys consisting of 3 attributes find closures of {EmployeeID, DepartmentID, ProjectID}, {EmployeeID, DepartmentID, ManagerID}, {EmployeeID, ProjectID, OfficeLocation} using the given algorithm

ChatGPT said:

Let's compute the closures of the following sets of attributes using the closure algorithm:

1. $\{EmployeeID, DepartmentID, ProjectID\}${EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ManagerID\}${EmployeeID,DepartmentID,ManagerID}
3. $\{EmployeeID, ProjectID, OfficeLocation\}${EmployeeID,ProjectID,OfficeLocation}

given-functional-dependencies-fd-4

Given Functional Dependencies (FD):

1. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}${EmployeeID,DepartmentID,ProjectID}
2. $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}${EmployeeID,Departm
3. $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}${ManagerID}→{EmployeeID,DepartmentID}
4. $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectID}

---

closure-of-employeeiddepartmentidprojectidemployeeiddepartmentidprojectid

1. Closure of $\{EmployeeID, DepartmentID, ProjectID\}^+${EmployeeID,DepartmentID,ProjectID}+ Closure of \{ EmployeeID,DepartmentID,ProjectID\}^{+}{EmployeeID,DepartmentID,Projec

1. **Initial result:** $\{EmployeeID, DepartmentID, ProjectID\}${EmployeeID,DepartmentID,ProjectID}
2. Apply FD $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID\}${EmployeeID,DepartmentID,

   - Result = $\{EmployeeID, DepartmentID, ProjectID, ManagerID\}${EmployeeID,DepartmentID,P

3. Apply FD $\{EmployeeID, DepartmentID, ProjectID\} \rightarrow \{ManagerID, OfficeLocation\}${EmployeeII
   again:

   - Result = $\{EmployeeID, DepartmentID, ProjectID, ManagerID, OfficeLocation\}${EmployeeID

4. Apply FD $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectID
   but these are already present in the result.

  - No new attributes are added.

5. Apply FD $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$ {ManagerID}→{EmployeeID,Department
   but these are already present in the result.

   - No new attributes are added.

- **Final result:** $\{EmployeeID, DepartmentID, ProjectID, ManagerID, OfficeLocation\}$ {EmployeeID

---

closure-of-employeeiddepartmentidmanageridemployeeiddepartmentidmanagerid

2. Closure of $\{EmployeeID, DepartmentID, ManagerID\}^+$ {EmployeeID,DepartmentID,ManagerID Closure of \{ EmployeeID,DepartmentID,ManagerID\}ˆ{+}{EmployeeID,DepartmentID,Mana

1. **Initial result:** $\{EmployeeID, DepartmentID, ManagerID\}$ {EmployeeID,DepartmentID,ManagerID}
2. Apply FD $\{ManagerID\} \rightarrow \{EmployeeID, DepartmentID\}$ {ManagerID}→{EmployeeID,Department
   but these are already present in the result.

   - No new attributes are added.

3. No other functional dependencies apply to this set.

- **Final result:** $\{EmployeeID, DepartmentID, ManagerID\}$ {EmployeeID,DepartmentID,ManagerID}

---

closure-of-employeeidprojectidofficelocationemployeeidprojectidofficelocation

3. Closure of $\{EmployeeID, ProjectID, OfficeLocation\}^+${EmployeeID,ProjectID,OfficeLocation}+
Closure of \{ EmployeeID,ProjectID,OfficeLocation\}^{+}{EmployeeID,ProjectID,OfficeLocat

1. **Initial result:** $\{EmployeeID, ProjectID, OfficeLocation\}${EmployeeID,ProjectID,OfficeLocation}
2. Apply FD $\{OfficeLocation\} \rightarrow \{EmployeeID, ProjectID\}${OfficeLocation}→{EmployeeID,ProjectID
   but these are already present in the result.

   - No new attributes are added.
3. No other functional dependencies apply to this set.

- **Final result:** $\{EmployeeID, ProjectID, OfficeLocation\}${EmployeeID,ProjectID,OfficeLocation}

---

summary-of-closures-3

Summary of Closures:

1. $\{EmployeeID, DepartmentID, ProjectID\}^+ = \{EmployeeID, DepartmentID, ProjectID, Manager$
   — contains **all** attributes.
2. $\{EmployeeID, DepartmentID, ManagerID\}^+ = \{EmployeeID, DepartmentID, ManagerID\}${Emplo
3. $\{EmployeeID, ProjectID, OfficeLocation\}^+ = \{EmployeeID, ProjectID, OfficeLocation\}${Employ

---

conclusion

Conclusion:

- The set $\{EmployeeID, DepartmentID, ProjectID\}${EmployeeID,DepartmentID,ProjectID} has a closure that contains all attributes, meaning it is a **candidate key**.
- The other sets, $\{EmployeeID, DepartmentID, ManagerID\}${EmployeeID,DepartmentID,ManagerID} and $\{EmployeeID, ProjectID, OfficeLocation\}${EmployeeID,ProjectID,OfficeLocation}, do not contain all attributes, so they are not candidate keys.

you-said-6

what are all the candidate keys that we have found so far?

ChatGPT said:

Based on the closures computed so far, the **candidate keys** are the sets of attributes whose closure contains **all** attributes. Let's summarize the findings:

candidate-keys-found

Candidate Keys Found:

1. $\{DepartmentID, OfficeLocation\}$**{DepartmentID,OfficeLocation}** — Contains all attributes.
2. $\{ProjectID, ManagerID\}$**{ProjectID,ManagerID}** — Contains all attributes.
3. $\{ManagerID, OfficeLocation\}$**{ManagerID,OfficeLocation}** — Contains all attributes.
4. $\{EmployeeID, DepartmentID, ProjectID\}$**{EmployeeID,DepartmentID,ProjectID}** — Contains all attributes.

summary-of-candidate-keys

Summary of Candidate Keys:

- **2-Attribute Candidate Keys:**
    1. $\{DepartmentID, OfficeLocation\}${DepartmentID,OfficeLocation}
    2. $\{ProjectID, ManagerID\}${ProjectID,ManagerID}
    3. $\{ManagerID, OfficeLocation\}${ManagerID,OfficeLocation}
- **3-Attribute Candidate Key:**
    1. $\{EmployeeID, DepartmentID, ProjectID\}${EmployeeID,DepartmentID,ProjectID}

These are all the **candidate keys** identified so far for the given functional dependencies.

you-said-7

will there be a 4 attribute candiate key that is minimal?

ChatGPT said:

No, there will not be a **4-attribute** minimal candidate key. Here's why:

definition-of-a-candidate-key

Definition of a Candidate Key:

A candidate key is a **minimal** set of attributes that can uniquely identify all other attributes in a relation. If you add more attributes to a candidate key, it is no longer minimal, meaning it's superfluous.

analysis

Analysis:

We have already found the following **minimal** candidate keys:

- **2-attribute candidate keys:**
  1. $\{DepartmentID, OfficeLocation\}$\{DepartmentID,OfficeLocation\}
  2. $\{ProjectID, ManagerID\}$\{ProjectID,ManagerID\}
  3. $\{ManagerID, OfficeLocation\}$\{ManagerID,OfficeLocation\}
- **1 3-attribute candidate key:**
  1. $\{EmployeeID, DepartmentID, ProjectID\}$\{EmployeeID,DepartmentID,ProjectID\}

Since these candidate keys are minimal, adding additional attributes to any of them would create a **superkey** (a superset of a candidate key), which by definition is not minimal.

conclusion-1

Conclusion:

There cannot be a 4-attribute **minimal** candidate key, as adding more attributes to any of the existing candidate keys would result in redundancy and violate the minimality condition. Thus, any candidate key with 4 attributes would not be minimal.

prompt-textarea

composer-background