MINIMAL COVER OR CANONICAL COVER OF FUNCTIONAL DEPENDENCY

Managing a large set of functional dependencies can result in unnecessary computational overhead. This is where the canonical cover becomes useful. The canonical cover of a set of functional dependencies F is a simplified version of F that retains the same closure as the original set, ensuring no redundancy.

An attribute in a functional dependency is considered extraneous if it can be removed without altering the closure of the set of functional dependencies.

Canonical Cover

A canonical cover is a set of functional dependencies that is equivalent to a given set of functional dependencies but is minimal in terms of the number of dependencies. Canonical Cover of functional dependency is also called minimal set of functional dependency or irreducible form of functional dependency. The process of finding the canonical cover of a set of functional dependencies involves the following steps:

Step 1: Combine Functional Dependencies with the Same Left-Hand Side

- If two or more functional dependencies in F have the same left-hand side, combine them into a single functional dependency by taking the union of their right-hand sides.
- **Example:**
    - A→B and A→C become A→BC.

Step 2: Eliminate Extraneous Attributes

An attribute is extraneous if removing it does not change the closure of the functional dependency set. There are two scenarios:

Extraneous Attributes on the Left-Hand Side:

For $X \rightarrow Y$, check if any attribute in X can be removed without affecting the closure.

To check:

- Remove an attribute A from X to form X'.
- Compute the closure of F with $X' \rightarrow Y$ instead of $X \rightarrow YX$.
- If the closure remains unchanged, A is extraneous.

Extraneous Attributes on the Right-Hand Side:

For $X \rightarrow Y$, check if any attribute in Y can be removed without affecting the closure.

To check:

- Remove an attribute B from Y.
- Compute the closure of F with $X \rightarrow Y'$, where Y' is Y without B.
- If the closure remains unchanged, B is extraneous.

Step 3: Decompose Functional Dependencies

If the right-hand side of a functional dependency has multiple attributes (e.g., $X \rightarrow AB$), decompose it into multiple functional dependencies, each with a single attribute on the right-hand side.

Example:

X→AB becomes X→A and X→B.

Step 4: Check for Redundant Dependencies

A functional dependency FD in F is redundant if it can be removed without changing the closure of F.

To check:

- Temporarily remove FD from F.
- Compute the closure of the remaining set.
- If the closure is the same as the closure of the original set, FD is redundant and can be removed.

Step 5: Verify the Final Canonical Cover

Ensure that each functional dependency is in its simplest form:

- The left-hand side has no extraneous attributes.
- The right-hand side contains only one attribute.

Check that the closure of the canonical cover is the same as the closure of the original set F.

Illustrative Examples

Example 1:

Consider a set of Functional dependencies: $F=\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$. Here are the steps to find the canonical cover –

Step 1: Combine Functional Dependencies with the Same Left-Hand Side

No two functional dependencies in F have the same left-hand side, so no changes are needed at this stage.

Step 2: Eliminate Extraneous Attributes

Check A→BC :

- The left-hand side A has no extraneous attributes because it's a single attribute.
- Check the right-hand side for extraneous attributes:
    - Split A→BC into A→B and A→C.
    - Now, F={A→B,A→C,B→C,AB→C}.

Check B→C :

- The left-hand side B has no extraneous attributes (it's a single attribute).
- No changes are needed.

Check AB→C :

- Checking $AB \rightarrow C$: First, check if A or B is extraneous.
- We can reach C without using $AB \rightarrow C$ with other functional dependencies; therefore, we remove $AB \rightarrow C$.
- Finally, we have $\{A \rightarrow B, A \rightarrow C, B \rightarrow C\}$.

Step 3: Decompose Functional Dependencies

All functional dependencies in F={A→B,A→C,B→C} have single attributes on the right-hand side, so no decomposition is needed.

Step 4: Check for Redundant Dependencies

Check A→C :

- Check each functional dependency to see if it can be reached without using it. For example, $A \rightarrow C$ can be reached with $A \rightarrow B$ and $B \rightarrow C$. Therefore, $A \rightarrow C$ is redundant and can be removed.
- Now F={A→B,B→C}.

Step 5: Final Canonical Cover

The final canonical cover is:

Fc={A→B,B→C}.

This is the simplified set of functional dependencies that has the same closure as the original set F.

Example 2:

Given F = { A → BC, B → C, A → B, AB → C }

- **Step 1 Reduction:** There are two functional dependencies with the same attributes on the left: A → BC, A → B are already in their simplest form.
- **Step 2 Elimination:** In A → BC, C is extraneous because A → C can be derived from A → B and B → C. Thus, we reduce it to A → B.
- **Step 3 Minimization:** No redundant dependencies remain.

Hence, the canonical cover is Fc = { A → B, B → C }

Example 3:

Given F = { A → BC, CD → E, B → D, E → A }

- **Step 1 Reduction:** Each left-hand side of the functional dependencies is unique and cannot be combined further.
- **Step 2 Elimination:** None of the attributes on the left or right sides of any functional dependency are extraneous.
- **Step 3 Minimization:** No dependencies are redundant.

Hence, the canonical cover is F = { A → BC, CD → E, B → D, E → A }.

How to Check Whether a Set of FDs F Canonically Covers Another Set of FDs G?

To verify whether a set of functional dependencies (F) canonically covers another set of functional dependencies (G), follow these steps:

Step 1: Compute the Closure of Each Set

Compute the closure of F:

- Use the attributes and dependencies in F to determine all the attribute sets that can be functionally determined.

Compute the closure of G:

- Similarly, calculate the attribute closures using the dependencies in G.

Step 2: Compare the Closures

For F to canonically cover G, the following conditions must hold:

The closure of F must be equivalent to the closure of G. That is, for every functional dependency in G, it must be derivable from F and vice versa.

Step 3: Derive Dependencies in G from F

For each functional dependency in G (e.g., $X \rightarrow Y$):

Compute $X+$ (closure of X) under F.

Verify that $Y \subseteq X+$.

If this is true for all functional dependencies in G, F covers G.

Step 4: Derive Dependencies in F from G

To ensure F and G are equivalent:

For each dependency in FF (e.g., $X \rightarrow Y$):

- Compute $X+$ (closure of X) under G.
- Check that $Y \subseteq X+$.

If all dependencies in F can be derived from G, the two sets are equivalent.

Step 5: Verify Minimality (Optional)

If F is already minimal (e.g., no extraneous attributes or redundant dependencies), and it satisfies the above steps, then F is a canonical cover of G.

Example:

Let $F=\{A \rightarrow B, B \rightarrow C\}$ and $G=\{A \rightarrow BC\}$.

1. **Compute Closure of F:**
    - $A+=\{A,B,C\}$ (using $A \rightarrow B$ and $B \rightarrow C$).
2. **Compute Closure of G:**

- A+={A,B,C} (using A→BC).

3. **Compare F with G:**
    - G can be derived from F: A→BC is equivalent to A→B and B→C.
    - F can be derived from G: A→B and B→C are derivable from A→BC.

Since F and G have the same closure and F is minimal, F canonically covers G.

Features of the Canonical Cover

- **Minimal:** The canonical cover is the smallest set of dependencies that can be derived from a given set of dependencies, i.e., it has the minimum number of dependencies required to represent the same set of constraints.
- **Lossless:** The canonical cover preserves all the functional dependencies of the original set of dependencies, i.e., it does not lose any information.
- **Deterministic:** The canonical cover is deterministic, i.e., it does not contain any redundant or extraneous dependencies.
- **Reduces Data Redundancy:** The canonical cover helps to reduce data redundancy by eliminating unnecessary dependencies that can be inferred from other dependencies.
- **Improves Query Performance:** The canonical cover helps to improve query performance by reducing the number of joins and redundant data in the database.
- **Facilitates Database Maintenance:** The canonical cover makes it easier to modify, update, and delete data in the database by reducing the number of dependencies that need to be considered.

Find the Minimal Cover

Given Functional Dependencies

- A -> B
- B -> C
- D -> ABC
- AC -> D

Step 1: Split the Functional Dependencies

- A -> B
- B -> C
- D -> A
- D -> B
- D -> C
- AC -> D

Step 2: Remove Redundant FDs

- A -> B (not redundant)
- B -> C (not redundant)
- D -> A (not redundant)
- D -> B (redundant, because D -> A and A -> B)
- D -> C (not redundant, as D -> B was removed)
- AC -> D (not redundant)

After removing redundancies FDs set became

- A -> B
- B -> C

- D -> A
- D -> C
- AC -> D

Step 3: Remove Extraneous Attributes

- In AC -> D, check if A or C is extraneous.
- Compute closures
- A+ = {A,B,C}
- C+ = {C}
- Since **A** alone does not give D, A is not extraneous
- Since **C** alone does not give D,C is not extraneous

So Minimal cover of (A -> B, B -> C, D -> ABC, AC -> D) => (A -> B, B -> C, D -> A, AC -> D)

By ensuring all functional dependencies are minimal and non-redundant, we obtain the correct minimal cover.