Closure Set

The closure set of a set F of Functional Dependencies (FDs) is the set of all FDs implied by F. This closure set is denoted by F⁺.

Armstrong’s Axioms

In 1974, William Ward Armstrong published a paper called *Dependency Structures of Data Base Relationships*. In this paper he presented what have become known as Armstrong’s Axioms – a set of rules that can be applied repeatedly to infer all the Functional Dependencies (FDs) implied by a set F of FDs. Using X, Y, and Z to denote sets of attributes over a relation schema R:

- Reflexivity: If X ⊇ Y (i.e. if X is a superset of Y), then X -> Y
- Augmentation: If X -> Y, then XZ -> YZ for any Z.
- Transitivity: If X -> Y and Y -> Z, then X -> Z

Armstrong’s Axioms are sound, in that they only generate FDs in F⁺ when applied to a set F of FDs. They are also complete in that repeated application of these rules will generate all FDs in the closure F⁺.

Additional rules

Using Armstrong’s axioms, additional rules can be created.

- Union: If X -> Y and X -> Z then X -> YZ
- Decomposition: If X -> YZ, then X -> Y and X -> Z

Trivial FD

A trivial FD is one in which the right side only contains attributes that also appear on the left side. These dependencies always hold due to reflexivity.
Examples:

If I have the following set F of FDs:

\[ F = \{A \rightarrow B, B \rightarrow C\} \text{ over set } \{ABC\}, \text{ then} \]
\[ F^+ = \{A \rightarrow A, B \rightarrow B, C \rightarrow C, AB \rightarrow AB, AC \rightarrow AC, BC \rightarrow BC, ABC \rightarrow ABC, \text{ (all from reflexivity)}, A \rightarrow B \text{ (given)}, B \rightarrow C \text{ (given)}, \text{ and } A \rightarrow C \text{ (transitivity), and } A \rightarrow BC \text{ (union)} \} \]

The trivial FDs are in the first set (i.e. all those that have the right side holding nothing but attributes on the left).

We derived \( A \rightarrow C \) using the transitivity rule from \( A \rightarrow B \) and \( B \rightarrow C \).

We derived the \( A \rightarrow BC \) using the union rule from \( A \rightarrow B \) and \( A \rightarrow C \).

Let’s look at another example – this one from the book:

Let’s assume we have the following schema:

Contracts (\texttt{ContractID, SupplierID, ProjectID, DeptID, PartID, Qty, Value})

For shorthand purposes let’s use:
\texttt{ContractID} = C
\texttt{SupplierID} = S
\texttt{ProjectID} = J
\texttt{DeptID} = D
\texttt{PartID} = P
\texttt{Qty} = Q and
\texttt{Value} = V.

1) Within this schema we know that \( C \rightarrow CSJDPQV \) from the fact that ContractID is the primary key of this relation.

2) We know that \( JP \rightarrow C \) from the fact that a project purchases a given part using a single contract. (business rule)

3) We know that \( SD \rightarrow P \) from the fact that a department purchases at most one part from a supplier. (business rule)

Ok – so given set \( F = \{C \rightarrow CSJDPQV, JP \rightarrow C, SD \rightarrow P\} \). Remember, the FDs are integrity constraints on the data defined by the business rules you are implementing.
What other ones do we know?

- We know \( \text{JP} \rightarrow \text{CSJDPQV} \) from transitivity
- We know \( \text{SDJ} \rightarrow \text{JP} \) from augmentation
- We know \( \text{SDJ} \rightarrow \text{CSJDPQV} \) from transitivity and the previous 2 rules
- We can decompose many additional FDs from the above as well, such as \( \text{C} \rightarrow \text{P} \), \( \text{C} \rightarrow \text{V} \), etc.

Computing the closure set of a large set of FDs can be VERY extensive. Instead we tend to look at the closure set of an attribute.

**Attribute Closure**

To identify the closure set of an attribute, we compute those attributes we can identify if we know the value of that attribute. Given the FDs in the previous example, the attribute closure of \( \text{C} \) (denoted as \( \text{C}^+ \)) is \{CSJDPQV\}. The attribute closure of \( \text{SD} \) (\( \text{SD}^+ \)) is \{SDP\}. 