**Example for Computing the Attribute Closure**

*Notation:* For the set \{A, C, E\} we permit to write \(ACE\) (juxtaposition) to be able to omit braces. In particular, \(\{D\}\) is written as \(D\).

*Example:* Let \(R(A, B, C, G, H, I)\) be a relation schema, and let \(F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CI \rightarrow G\}\) be a set of FDs.

**Task:** Compute \(AG^+\).

**Solution:** We use the algorithm *AttrClosure* and initialize \(AG^+\) with \(AG\).

In the loop we set \(Old_AG^+ := AG\) and check all FDs whether they can contribute to \(AG^+\).

First we take \(A \rightarrow B\) and check whether \(A \subseteq AG^+\) holds. This is the case. Therefore, we set \(AG^+ := AG^+ \cup B = ABG\) (due to transitivity).

Next, we take \(A \rightarrow C\). Using the same argument as before, we obtain \(AG^+ := AG^+ \cup C = ABCG\).

Next, we take \(CG \rightarrow H\). We find that \(CG \subseteq AG^+\) holds. We get \(AG^+ := AG^+ \cup H = ABCGH\).

Next we take \(CI \rightarrow G\). We find that \(CI \not\subseteq AG^+\) holds.

Since \(Old_AG^+ \neq AG^+\) holds, we perform a second loop. We set \(Old_AG^+\) to \(AG^+\), that is, \(Old_AG^+ := ABCGH\). We see soon that no FD from \(F\) can increase \(AG^+\). This means that \(Old_AG^+ = AG^+\) holds, the algorithm terminates, and we get \(AG^+ := ABCGH\).
Canonical cover

- In general, distinct equivalent sets of FDs exist. Two sets \( F \) and \( G \) of FDs are called **equivalent** iff \( F^+ = G^+ \) holds.

- Definition of equivalence is convincing, because the equality of the closures for \( F \) and \( G \) implies that the same FDs can be inferred from \( F \) and \( G \).

- For a given set \( F \) of FDs there exists a unique closure \( F^+ \).

- **drawbacks of the closure** \( F^+ $$:
  - in general very many FDs in \( F^+ \) so that the handling with \( F^+ \) becomes difficult
  - large redundant set of FDs that has to be checked as consistency tests for database modifications

- **goal**: computation of a most possible small set of FDs which are equivalent to \( F $$
  \rightarrow$$ less effort for testing whether a new or updated tuple violates a FD
$F_c$ is called **canonical cover** of a given set $F$ of FDs, if holds:

- $F_c^+ = F^+$
- In $F_c$ there are no FDs $A \rightarrow B$ where $A$ or $B$ contain *extraneous* attributes, i.e., they are reduced as much as possible.

We cannot omit any attribute on the **left** sides of any FD, otherwise we would change the semantics:

$$\forall a \in A : (F_c - \{A \rightarrow B\} \cup \{(A - \{a\}) \rightarrow B\})^+ \neq F_c^+$$

*Example*: schema `supplier(sname, saddr, product, price)` and FDs `{sname, product} \rightarrow \{saddr\}` and `{sname, product} \rightarrow \{price\}`. Can we omit one of the attributes on the left sides?

We cannot omit any attribute on the **right** sides of any FD, otherwise we would change the semantics:

$$\forall b \in B : (F_c - \{A \rightarrow B\} \cup \{A \rightarrow (B - \{b\})\})^+ \neq F_c^+$$

- Each left side of the FDs in $F_c$ occurs only once, i.e.,

  if $A \rightarrow B$ and $A \rightarrow C$ hold, then in $F_c$ only the FD $A \rightarrow B \cup C$ is used.
algorithm for computing the canonical cover

- **step 1:** For each FD $A \rightarrow B \in F$ perform a left reduction: check for all $a \in A$ whether the attribute $a$ is extraneous, i.e., whether
  
  \[ B \subseteq \text{AttrClosure}(F, A \setminus \{a\}) \]
  
  holds. If this is the case, replace $A \rightarrow B$ by $(A \setminus \{a\}) \rightarrow B$.

- **step 2:** For each remaining FD $A \rightarrow B \in F$ perform the right reduction: check for all $b \in B$, whether the attribute $b$ is extraneous, i.e., whether
  
  \[ b \in \text{AttrClosure}(F \setminus \{A \rightarrow B\} \cup \{A \rightarrow (B \setminus \{b\})\}, A) \]
  
  holds. If this is the case, replace $A \rightarrow B$ by $A \rightarrow (B \setminus \{b\})$.

- **step 3:** Remove the FDs of the form $A \rightarrow \emptyset$ which perhaps have been produced in the previous step.

- **step 4:** By using the union rule replace all FDs of the form $A \rightarrow B_1, \ldots, A \rightarrow B_n$ by

  \[ A \rightarrow B_1 \cup \ldots \cup B_n \]
example
- Given the set \( F = \{A \rightarrow B, B \rightarrow C, A \cup B \rightarrow C\}. \)
- step 1: \( A \cup B \rightarrow C \) is replaced by \( A \rightarrow C \), because \( B \) on the left side is extraneous (\( C \) is already functionally dependent from \( A \) by the first two FDs).
- step 2: \( A \rightarrow C \) is replaced by \( A \rightarrow \emptyset \), because \( C \) on the right side is extraneous. This results from the fact that \( C \subseteq \text{AttrClosure}(\{A \rightarrow B, B \rightarrow C, A \rightarrow \emptyset\}, A) \).
- step 3: \( A \rightarrow \emptyset \) is removed. We obtain: \( F_c = \{A \rightarrow B, B \rightarrow C\}. \)
- step 4: Nothing to be done.
Decomposition of a relation schema

- **Normalization**: In order to eliminate anomalies (redundancies, update, insertion and deletion anomalies), the relation schema $R$ is decomposed into $n$ relation schemas $R_1, ..., R_n$.

- two fundamental correctness criteria for such a decomposition:
  - **losslessness (lossless join decomposition)**: An arbitrary instance $r(R)$ must be reconstructable from the instances $r_1(R_1), ..., r_n(R_n)$.
  - **dependency preservation**: All FDs which hold for schema $R$ must be transferable to the schemas $R_1, ..., R_n$ and must be efficiently checkable.

- **losslessness**
  - It is sufficient to confine oneself to the decomposition of $R$ into two relation schemas $R_1$ and $R_2$.
  - Of course, we must require: $R = R_1 \cup R_2$.
  - A decomposition of $R$ into $R_1$ and $R_2$ is **lossless** if for all relations $r(R)$ holds:
    \[ r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r). \]
  - That is, reconstruction must be possible by natural join.
criteria for the losslessness of a decomposition

Let $R$ be a relation schema and $F_R$ the set of FDs. A decomposition of $R$ in $R_1$ and $R_2$ is lossless, if

$$(R_1 \cap R_2) \rightarrow R_1 \in F_R^+ \quad \text{or} \quad (R_1 \cap R_2) \rightarrow R_2 \in F_R^+$$

i.e., $R_1 \cap R_2$ is a superkey for $R_1$ or $R_2$

Alternative formulation: Let $R = A \cup B \cup C$, $R_1 = A \cup B$ and $R_2 = A \cup C$ with pair-wise disjoint attribute sets $A$, $B$, $C$. Then:

$B \subseteq \text{AttrClosure}(F_R, A) \quad \text{or} \quad C \subseteq \text{AttrClosure}(F_R, A)$

must hold.

sufficient, but not necessary condition for losslessness

In a relation schema $R$, $X \subseteq R$ is called a superkey, if $X \rightarrow R$ holds.

example for a lossy decomposition:

- The decomposition of the relation $R$(sname, saddr, product, price) in the two relations supplier(sname, saddr, product) and offer(product, price) is not lossless, since in general $R \neq \text{supplier} \bowtie \text{offer}$ holds.

- reasons:
  - Product does not functionally determine the price.
  - Product does not functionally determine supplier’s name and address.
Lossless Join Decomposition

- Assume we decompose a relation $R$ into relations with sets of attributes $S_1, S_2, \ldots, S_k$.
- This decomposition is lossless if $R$ can be reconstructed, that is, it holds that
  $$ \pi_{S_1}(R) \Join \pi_{S_2}(R) \Join \ldots \Join \pi_{S_k}(R) = R $$
- Note that this means that $S_i \cap S_{i+1} \neq \emptyset$ for all $1 \leq i < k$. Otherwise, we cannot compute a natural join.
- Example of a lossy decomposition

\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
4 & 2 & 5 \\
\hline
\end{array}
\quad R_1 = \pi_{A,B}(R) =
\begin{array}{|c|c|}
\hline
A & B \\
\hline
1 & 2 \\
4 & 2 \\
\hline
\end{array}
\quad R_2 = \pi_{B,C}(R) =
\begin{array}{|c|c|}
\hline
B & C \\
\hline
2 & 3 \\
2 & 5 \\
\hline
\end{array}
\]\n
\[
R_1 \Join R_2 = 
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
1 & 2 & 5 \\
4 & 2 & 3 \\
4 & 2 & 5 \\
\hline
\end{array} \neq R
\]

The tuples $(1, 2, 5)$ and $(4, 2, 3)$ are “too much”. The deeper reason is that neither $B \rightarrow A$ nor $B \rightarrow C$ holds.