Schlüssel

Online-Tools zur Bestimmung von Schlüssel

- DB->normalizer der TU München
- Normalization Tool der Universität Griffith, Australien

Definitionen der wichtigsten Begriffe

Superschlüssel (gelegentlich auch Oberschlüssel genannt) Attribut oder Attributkombination, von der *alle Attribute* einer Relation funktional *abhängen*.[1] alle Attribute abhängen

Schlüsselkandidat (auch *Kandidatenschlüssel* oder Alternativschlüssel genannt) ist ein *Minimaler* Superschlüssel. Keine Teilmenge dieses Superschlüssels ist ebenfalls Superschlüssel. Kandidatenschlüssel Minimaler

Primärschlüssel *Unter allen Schlüsselkandidaten* einer Relation wird ein sogenannter Primärschlüssel *ausgewählt*. Unter allen Schlüsselkandidaten ausgewählt

Schlüsselattribut Attribut, das *Teil eines Schlüsselkandidaten* ist. Teil eines Schlüsselkandidaten

Nicht-Schlüsselattribut Attribut, das an *keinem* der Schlüsselkandidaten beteiligt ist.[2] keinem

Bestimmung von Schlüsselkandidaten ohne Algorithmus

Schlüsselkandidaten können manchmal auch ohne Anwendung des Algorithmus gefunden werden. Bestimmen Sie die Menge X= A 1 , ..., A n der Attribute, die auf keiner rechten Seite einer funktionalen Abhängigkeit aus F (Menge der funktionalen Abhängigkeiten) vorkommen. Diese Attribute kann man nicht durch Ableiten gewinnen. Sie müssen in jedem Schlüsselkandidaten vorhanden sein sein. Auch Attribute, die in keiner der Funktionalen Abhängigkeiten in irgendeiner Weise vorkommen, müssen Teil jedes Schlüsselkandidaten sein.

Bestimmen Sie dann die Attributhülle von X, um zu prüfen, ob X Superschlüssel von R ist. Falls ja, ist X einziger Schlüsselkandidat, da man alle darin enthaltenen Attribute nicht aus Ableitungen gewinnen kann und weil die Schlüsselkandidaten minimal sind. Falls nein, muss entweder der Algorithmus zur Bestimmung von Schlüsselkandidaten angewendet werden oder man findet die Schlüsselkandidaten durch „systematischen“ Aufbau, ausgehend von der Menge X.[3]

[1] Kemper und Eickler, *Datenbanksysteme*, Seite 181 Kapitel 6.2 „Superschlüssel“.
[2] *Qualifizierungsmaßnahme Informatik - Datenbanksysteme 4*, Seite 7.
[3] *Qualifizierungsmaßnahme Informatik - Datenbanksysteme 4*, Seite 11.

Zusammenfassung

keiner rechten Seite

- Attribute, die auf *keiner rechten Seite* einer Funktionalen Abhängigkeit vorkommen.

keiner Funktionalen Abhängigkeit

- Attribute, die in *keiner Funktionalen Abhängigkeit* vorkommen.

Algorithmus zur Bestimmung von Schlüsselkandidaten[4]

Algorithmus 1: Algorithmus zur Bestimmung von Schlüsselkandidaten

begin
  $Erg \longleftarrow \{\}$;
  $Test \longleftarrow \{\{$ *alle Attribute der Relation* $\}\}$;
  $K \longleftarrow \{$*alle Attribute der Relation*$\}$;
  while *bis Test leer ist* **do**
    `/* Wähle ein Attribut aus` K `*/`
    foreach $A \longleftarrow K$ **do**
      `/* Also die Menge` K `ohne das Attribut` A `*/`
      if $AttrHülle(F, K \backslash \{A\}) = R$ **then**
        streiche K aus *Test*;
        füge $K \backslash \{A\}$ in *Test* ein;
        `/*` K `selbst bleibt unverändert */`
      end
    end
  end
  entferne ein anderes Attribut aus K, so lange, bis alle Attribute reihum untersucht wurden.;
  wenn kein $K\{A\}$ Superschlüssel $\Rightarrow$ K ist Schlüsselkandidat! Füge K zu *Erg* hinzu und lösche K aus *Test*.;
  mache dasselbe mit allen Mengen, die jetzt in *Test* sind, bis *Test* leer ist;
end

Algorithmus zur Bestimmung aller Superschlüssel in Java

```
180  /**
181   * Berechne alle Superschlüssel (einschließlich der Kandidatenschlüssel).
182   *
183   * @param attribute       Eine Menge aus Attributen.
184   * @param abhaengigkeiten Eine Menge aus Funktionalen Abhängigkeiten.
185   *
186   * @return Eine Menge aus Superschlüsseln. Jeder Superschlüssel ist wiederum
187   *         eine Menge aus Attributen.
188   */
```

[4] *Qualifizierungsmaßnahme Informatik - Datenbanksysteme 4*, Seite 8.

```java
public static Set<Set<Attribut>> berechneSuperSchlüssel(Set<Attribut>
 ↪  attribute, Set<Abhaengigkeit> abhaengigkeiten) {
  Set<Set<Attribut>> schlüssel = new HashSet<>();
  if (attribute.isEmpty()) {
    for (Abhaengigkeit abhaengigkeit : abhaengigkeiten) {
      attribute.addAll(abhaengigkeit.left);
      attribute.addAll(abhaengigkeit.right);
    }
  }
  Set<Set<Attribut>> potenzMenge = erzeugeReduziertePotenzMenge(attribute);
  for (Set<Attribut> attributMenge : potenzMenge) {
    if (berechneAttributHülle(attributMenge,
     ↪  abhaengigkeiten).equals(attribute)) {
      schlüssel.add(attributMenge);
    }
  }
  return schlüssel;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/db/AlgorithmenSammlung.java`

Algorithmus zur Bestimmung aller Schlüsselkandidaten in Java

```java
/**
 * Berechne alle Schlüsselkandidaten (bzw. Kandidatenschlüssel).
 *
 * @param attribute       Eine Menge aus Attributen.
 * @param abhaengigkeiten Eine Menge aus Funktionalen Abhängigkeiten.
 *
 * @return Eine Menge aus Schlüsselkandidaten. Jeder Schlüsselkandidate besteht
 *         wiederum aus einer Menge von Attributen.
 */
public static Set<Set<Attribut>> berechneKandidatenSchlüssel(Set<Attribut>
 ↪  attribute,
    Set<Abhaengigkeit> abhaengigkeiten) {
  Set<Set<Attribut>> superSchlüssel = berechneSuperSchlüssel(attribute,
   ↪  abhaengigkeiten);
  Set<Set<Attribut>> zuEntfernen = new HashSet<>();
  for (Set<Attribut> schlüssel : superSchlüssel) {
    for (Attribut attribut : schlüssel) {
      Set<Attribut> verbleibende = new HashSet<>(schlüssel);
      verbleibende.remove(attribut);
      if (superSchlüssel.contains(verbleibende)) {
        zuEntfernen.add(schlüssel);
        break;
      }
    }
  }
  superSchlüssel.removeAll(zuEntfernen);
  return superSchlüssel;
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/db/AlgorithmenSammlung.java`

Literatur

[1] Alfons Kemper und André Eickler. *Datenbanksysteme. eine Einführung*. 2013.

[2] *Qualifizierungsmaßnahme Informatik - Datenbanksysteme 4. Funktionale Abhängigkeiten, Normalformen, Kanonische Überdeckung, Synthesealgorithmus*. `https://www.studon.fau.de/file2480907_download.html`.