Datenmodellierung Quick Help

Bsp 1 a:

Schauen welcher Buchstabe **rechts nicht steht**=> sicher Teil des „Schlüssels“.

Da aber oft dieser „Schlüssel“ nicht reicht, da man auch noch andere Buchstabend braucht um alle Buchstaben zu erreichen, muss man **überprüfen ob und was man noch braucht**.

Das macht man indem man schaut, ob man mit dem „Schlüssel“ alleine durch kommt oder nicht. Wenn man durch kommt, ist der „Schlüssel“ auch der richtige Schlüssel. Wenn man nicht durch kommt, ist der „Schlüssel“ nur ein Teilschlüssel, und es fehlt noch ein Buchstabe.

Deswegen muss man dann zu dem „Schlüssel“ den man gefunden hat, durch probieren herausfinden welcher Buchstabe man noch braucht sodass der „Schlüssel“ ein richtiger Schlüssel ist.

Wenn man als Schlüssel zum Beispiel B hat, und mit B alleine nicht durch kommt, muss man dazu eben einen Buchstaben mehr anhängen. Sprich BA, BB, BC, BD, BE, BF … da aber BB keinen Sinn macht scheidet das schon ein Mal aus. Über bleiben also nur mehr BA, BC, BD, BE, BF. Nun probiert man aus, mit welchen von diesen möglichen Schlüsseln man als Ziel kommt, und wenn es mehrere sind, **schreibt man ALLE auf als Schlüssel**!

Wichtig ist generell bei dem ganzen Beispiel, dass wenn links 2 oder mehr Buchstaben stehen, muss man ALLE haben damit man auf das weiter gehen kann was rechts steht.

Bei **„Zerlegung in 3NF“** wird angeben **wie man alle Buchstabend „abklappert“**. Sprich mit dem ersten Ausdruck beginnen und schauen wie man durch kommt. In der Regel verwendet man, wenn man mehrere Schlüssel hat den, der alphabetisch zuerst kommt.

Wenn man gleiches hat muss man streichen wie z.b. E zu F und F zu E=> reicht eines von beiden anzuschreiben.

Nicht vergessen das man das was links steht unterstreicht bzw den ganzen Schlüssel im Ergebnis (und man vor allem den dann auch dort **noch Mal am Schluss angeben** muss). Man kann =>A weglassen, genauso wenn es Schlüssel ist, man es dann am Schluss nicht nochmal angeben muss!

Bsp 1 b:

Schlüssel suchen (wie oben):Schauen was rechts fehlt bzw. wenn es viele Ausdrücke sind, probieren mit welchen Buchstaben man startet und vollständig durchkommt. Wenn es mehrere gibt, die einfach mit Beistrichen getrennt ALLE angeben!

Dann muss man **schauen, ob bei den Ausdrücken der Angabe auf JEDEN der Ausdrücke** (es muss aber immer nur einer der Eigenschaften zutreffen) **einer der folgenden Eigenschaften zutrifft** auf Grund der Schlüssel:

1) FD: Trivial. Ist gegeben wenn rechts das steht was auch (unter anderem)links steht.
Zum Beispiel AB=>B, FA=>A, BC=> B

2) rechts Schlüssel Attribut/ Teil von Schlüssel.

Zum Beispiel AC Schlüssel und rechts würden A oder C stehen

3) links Oberschlüssel: links steht GENAU der Schlüssel drinnen, nicht mehr, nicht weniger!

Zum Beispiel Schlüssel ist AC und es steht in Angabe AC=>irgendwas oder DE ist Schlüssel und DE=> irgendwas. NICHT gehen würde es bei AC ist Schlüssel und ABC=> irgendwas, da eben dort mehr als der Schlüssel steht!

Damit 3NF zutrifft, darf man 1, 2 und 3 verwenden.
Damit BCNF zutrifft, darf man 1 und 3 verwenden, also NICHT 2!

Nun schaut man also ob jeder Ausdruck der Angabe zumindest eines der Eigenschaften die man verwenden darf erfüllt. **Wenn Ja, geht es und man kreuzt JA an. Gibt es aber auch nur bei einem einzigen Ausdruck keine gültige Eigenschaft, ist es NEIN!**

Beispiel 1:
AC=> BDF
F=> A
D=> E

Schlüssel: AC, FC //DC geht nicht!
AC=> BDF: 3, da links genau Schlüssel AC steht.
F=> A: 2, da A Teil des Schlüssels AC ist.
D=> E: **keine Eigenschaft trifft zu=> NEIN ankreuzen**

Beispiel 2:
CD=> AB
AE=> CDF
AB=> AE

Schlüssel: CD, AB, AE
CD=> AB: 2 (da AB im Schlüssel AB ist) und 3 (da links genau Schlüssel CD steht)
AE=> CDF: 2 (da CD im Schlüssel CD ist) und 3 (da links genau Schlüssel AF steht)
AB=> AE: 2 (da AE im Schlüssel AE ist) und 3 (da links genau Schlüssel AB steht)
Alle Eigenschaften erfüllt für 3NF und BCNF=> Beides JA

Beispiel 3:
AB=> CDE
AE=> F
E=> B

Schlüssel: AE, AB
AB=> CDE: 2 (da E im Schlüssel AE ist) und 3 (da links genau Schlüssel AB steht)
AE=> F: 3 (da links genau Schlüssel AE steht)
E=> B: 2 (da B im Schlüssel AB ist)

Da wir aber bei **BCNF aber nur 1 und 3 verwenden dürfen**, beim Ausdruck E=> B **aber nur die 2 Eigenschaften erfüllen, ist es NICHT BCNF, sondern nur 3NF** (da alle Ausdrücke entweder 1, 2 oder 3 haben)=> Nein bei BCNF und JA bei 3NF.

Bsp 2 (Quelle: http://www.informatik-forum.at/showthread.php?p=387344):

Angabe: F={AB->CD, A->E, E->D, AD->AF}

1.Rechtsreduktion
Wo immer auf der rechten Seite mehr als 1 Buchstabe vorkommt, aufspalten:
Das ist bei AB->CD und bei AD->AF der Fall und wird daher aufgespalten in AB->C, AB->D, AD->A und AD->F.

Wir haben also: F' = {AB->C, AB->D, A->E, E->D, AD->A, AD->F}

2. Linksreduktion
Schauen, ob auf der linken Seite mehr als 1 Buchstabe vorkommt. Das ist bei uns bei AB->C, AB->D, AD->A und AD->F der Fall.

3. Attributhülle von allen diesen Buchstaben auf der linken Seite erstellen:
Erstelle Attributhülle von A:
1. A ist auf alle Fälle immer erreichbar, wenn A gegeben ist, daher ist A in der Attributhülle.
2. Der nächste nur durch A erreichbare Buchstabe ist E (A->E). E wird zur Attributhülle gegeben
3. Nachdem E jetzt vorhanden ist, kann ich auch D erreichen (E->D). D zur Attributhülle hinzufügen.
4. Da A und D vorhanden sind, kann nun auch F hinzugefügt werden (AD->F).
Also Attributhülle von A ist somit AEDF

Erstelle Attributhülle von B:
1. B ist auf alle Fälle immer erreichbar, wenn B gegeben ist, daher ist B in der Attributhülle.
2. Sonst ist nichts mehr erreichbar.
Attributhülle von B ist somit B

Erstelle Attributhülle von D:
1. D ist auf alle Fälle immer erreichbar, wenn D gegeben ist, daher ist D in der Attributhülle.
2. Sonst ist nichts mehr erreichbar.
Attributhülle von D ist somit D

Ergebnis:
A+={A, E, D, F}
B+={B}
D+={D}

Betrachten wir nun AB->C:
a) Ist A in AB->C überflüssig? Nein, da C nicht in der Attributhülle von B liegt
b) Ist B in AB->C überflüssig? Nein, da C nicht in der Attributhülle von A liegt

AB->D
a) ist A überflüssig? Nein, weil D nicht in Attr.h. von B vorkommt.
b) ist B überflüssig? Ja, weil D in Attr.h. von A liegt

AD->A
a) ist A überflüssig? Nein, weil A nicht in Attributhülle von D liegt
b) ist D überflüssig? Ja, weil A in Attr.hülle von A liegt

AD->F
a) ist A überflüssig? Nein, weil F nicht in der Attributhülle von D liegt

b) ist D überflüssig? Ja, weil F in Attributh. von A liegt

Wir erhalten also: F'' = {AB->C, A->D, A->E, E->D, A->A, A->F}

4. Redundanzen entfernen
wir betrachten nacheinander alle FDs, und schauen, ob man irgendwelche weglassen kann.

AB->C
Attributhülle von AB bilden, wobei AB->C für die Bildung der Attributhülle nicht mitberechnet werden darf. Also AB+ (von F'' ohne AB->C) = {ABDEF} da C nicht in der Attributhülle liegt, kann diese FD nicht weggestrichen werden.

A->D
A+ (von F'' ohne A->D) = {AEDF}
Da D in der Attributhülle liegt, kann die FD weggelassen werden Wir haben also nun:
F'' = {AB->C, A->E, E->D, A->A, A->F}

A->E
A+(von F'' ohne A->E) = {AF}
FD kann nicht weggelassen werden, da E nicht in der Attributhülle liegt

E->D
E+(von F'' ohne E->D) = {E}
FD kann nicht weggelassen werden, da D nicht in der Attributhülle liegt

A->A
A+(von F'' ohne A->A) = {AEDF}
FD kann weggelassen werden, da A in Attributhülle liegt

F'' = {AB->C, A->E, E->D, A->F}

A->F
A+(von F'' ohne A->F) = {AED}
FD kann nicht weggelassen werden, da F nicht in der Attributhülle liegt

5. Zusammenfassen
Alle FDs die den- bzw. dieselben Buchstaben auf der linken Seite haben zusammenführen:

F''' = {AB->C, A->EF, E->D}

Fertig!

Bei mir hat diese Methode bei allen Beispielen bis jetzt funktioniert. Was vielleicht noch zu erwähnen ist: Wenn bei der Linksreduktion auf der linken Seite z.B. 3 Buchstaben stehen, dann nicht von jedem einzelnen Buchstaben eine Attributhülle bilden, sondern zuerst einmal von jeweils 2. Z.B. ABC->D

Dann Attributhülle von AB, BC, AC bilden. Da kann man dann schon einmal schauen, ob ein Buchstabe überflüssig ist. (A ist überflüssig, wenn D in Attributhülle von BC liegt, B ist überflüssig, wenn D in Hülle von AC liegt usw.)

Sollte man in diesem Schritt eine Überflüssigkeit feststellen, z.B. es genügt AC->D, dann kann man

jetzt von A und C die Attributhülle erstellen, und schauen, ob man nicht noch einen Buchstaben wegtun kann.

Bsp 5 (gut zum Thema: http://www.informatik-forum.at/showthread.php?t=32374):

Zeichen	Min	Max
∪ (Prim)	Wert vom Primär Schlüssel	Beide Werte zusammen zählen
∪ (Sek)	1 (wenn alle gleich=> mind. 1 Tupel)	Beide Werte zusammen zählen
Nat. Join (gemeinsames)	0	Siehe weiter unten!
Nat. Join (haben nichts gemeinsam zum joinen)	Kreuzprodukt (Beides multiplizieren)	Kreuzprodukt
R.Out.Join	0	Siehe weiter unten!
σ	0	Wert von Relation
σ Prim= 1	0	1
σ sek= 1	0	Tupelanzahl
−	Großer Wert – kleinem Wert	Großer Wert
(R Join S)–(R.Out.Join S)	0	0
∩	0	kleinster Wert von beiden
A ∩ A	TupelwertA	TupelwertA
X	TupelwertA*TupelwertB	TupelwertA*TupelwertB

Beim Natural Join und Right Outer Join schaut man, welche gemeinsamen Schlüssel sie haben. Danach schaut man ob bei einen der „Tabellen" dieser Schlüssel ein Primärschlüssel ist, und wenn ja, nimmt man beim Max die Tupelanzahl von der „Tabelle" bei der es NICHT Primärschlüssel ist.

Generell muss man auf passen, ob es sich um einen Primärschlüssel (Prim, bei der Angabe zu erkennen da er unterstrichen ist) handelt oder nicht. Primärschlüsseln sind IMMER unterschiedlich, während Sekundärschlüssel verschieden sein KÖNNEN, aber nicht MÜSSEN. Man kann sich also bei Sekundärschlüssel aussuchen, entsprechend ob man Min oder Max haben will, ob sie gleich sind oder nicht!

Pi: verhält sich wie select distinc => Duplikate werden heraus gelöscht.
$P_{E<=D}$ T: Man hat in T ein D, aber kein E drinnen. Aber man will es zum Beispiel mit einer anderen Tabelle vereinigen und dafür das D in ein E „umbenennen", sodass das D als E mit dem Inhalt von E einer anderen Tabelle vereinigt wird.

Allgemeines

σ Selektion =>Zeile auswählen
$\sigma_{Semester>10}(Student)$… liefert alle Zeilen zurück, wo ein Student mehr als 10 Semester braucht.

π Projektion =>Spalte auswählen
$\pi_{Rang}(Professor)$… liefert alle Spalten zurück, wo der Rang der Professoren drinnen steckt.

∪ Vereinigung =>Zusammen nehmen
$(Assistent) \cup (Professor)$… liefert alle Spalten zurück, wo der Rang der Professoren drinnen steckt.

− Mengendifferenz =>Abziehen
$\pi_{MatrNr}(Student) - \pi_{MatrNr}(prüfen)$ … liefert alle Studenten zurück, die noch keine Prüfung gemacht haben.

X Kreuzprodukt/ Kartesisches Produkt =>Multiplizieren
$R\ x\ S$ … liefert alle möglichen Paaren Tupeln R*S zurück

Natürlicher Join=> Kreuzprodukt – Doppelten
R = m+k S = n+k => R natJoin S = m + n + k (bzw (R-S) + (R ∩ S) + (S - R))
(Student natJoin hören) natJoin Vorlesung … listet alle Studenten auf und welche Vorlesungen sie hören, wobei z.b. Mehrfachnennungen von MatrNr wegfallen.

Version: 0.8
Zusammenfassung: Martin Tintel (mtintel)

Bei Fragen/ Fehler einfach eine Mail an mtintel@gmx.at schicken.