Version: 0.4

Bei Fragen unter http://www.informatik-forum.at/showthread.php?t=49930 stellen oder per Mail an mtintel@gmx.at

Bsp 1:

Bei Weak Entities muss man sich von der Entität, an die sie „angeknüpft“ ist den Schlüssel holen.

Bei Beziehungen mit Attribut das Attribut anschreiben und die Schlüssel von „benachbarten“ Entitäten.

Generailsierung: Schlüssel zusätzlich vom Allgemeinen zum Speziellen geben
[1,1] links und [1,*] bzw [0,* ???] rechts: Auf der 1,1 Seite die Schlüssel vom 1,* zusätzlich dazu geben
[0,1] links und [1,*] rechts=> Tabelle machen und von Links Schlüssel als Primär nehmen und von rechts als Sekundär
[1,*] links und [1,*] rechts => Tabelle machen und Schlüssel aller beteiligten Relationen als Primär Schlüssel hinzufügen

Quelle: http://www.informatik-forum.at/showpost.php?p=387365&postcount=53

Ich schaue mir **zuerst alle Entities an und füge mal die angegebenen Attribute ein**. Also:
L(ID1, Attr2)
K(ID1, ID2)
JJ(Attr2, Attr3)
M(Attr1, ID2)
J(Attr1, ID1)

Jetzt schaue ich mir die Generalisierungen an
Da gibt es eine zw. JJ und J. Da JJ Spezialfall von J ist, erbt JJ alle Schlüssel von J und diese sind bei JJ auch Schlüssel. Daher:

JJ(Attr2, Attr3, J.ID1)

Nächster Schritt: **Relationen auflösen**
Rel-D:
[1,1] auf Seite von L und [1,*] auf der Seite von K. Schlüssel von K werden zu L als Attribute hinzugefügt:
L(ID1, Attr2, K.ID1, K.ID2)

Warum? Weil jedem L genau ein K zugeordnet ist [1,1]. Umgekehrt ginge es nicht: L-Schlüssel können nicht als Attribute K hinzugefügt werden, da jedem K ein oder mehere L's zugeordnet sind.

Rel-F:
[1,1] auf Seite von J und [1,*] auf der Seite von K. Selber Fall wie bei Rel-D: Schlüssel von K werden zu L als Attribute hinzugefügt:

J(Attr1, ID1, K.ID1, K.ID2)

Rel-E:
[1,1] auf Seite von M und [1,*] auf der Seite von J. Ähnlicher Fall wie bei Rel-D bzw. Rel-F: Schlüssel von J werden zu M hinzugefügt, aber mit dem Unterschied, dass sie in M nun auch Schlüssel sind, da

es sich bei M um eine Weak-Entity handelt. Also: M(Attr1, ID2, J.ID1)

Rel-B:
[0,1] auf der Seite von L und [1,*] auf der Seite von M. L-Schlüssel können auf gar keinen Fall bei M als Attribute dazugeschrieben werden, weil einem M ein oder mehere L zugeordnet sind. Schauen wir uns also die andere Seite an. Auch M-Schlüssel können L nicht zugeordnet werden, da unsere Datenbank keine Nullwerte akzeptiert - denn jedem L ist kein oder genau ein M zugeordnet(wäre dies anders, dann wäre es möglich). Also bleibt uns nichts anderes übrig, als für diese Relation eine eigene Tabelle zu machen. Prinzipiell sind immer alle Schlüssel der beteiligten Entities in dieser eigenen Tabelle vorhanden also L.ID1, M.ID2, M.J.ID1. Die Frage ist nun noch was Schlüssel ist. Es reicht in diesem Fall L.ID1 zum Schlüssel zu machen, da wir eine [0,1] Beziehung auf der L-Seite haben, d.h. wenn einem L ein M zugeordnet ist, dann kann es nur eins sein und nicht mehrere.
Wir haben also am Ende:

Rel-B(L.ID1, M.ID2, M.J.ID1)
Rel-A:
Auf der einen Seite (sagen wir JJ2) [1,*] und auf der anderen Seite (sagen wir JJ1) [0,1]. Selber Fall wie bei Rel-B, also:

Rel-A(JJ1.ID1, JJ2.ID1)
Rel-C:
Auf L-, K- und J-Seite [1,*] Beziehungen. Das heißt wir müssen aus dieser Relation eine eigene Tabelle machen. Schlüssel aller beteiligten Relationen als Schlüssel hinzufügen (weil bei jeder der entities gilt 1 oder *) plus das Zusatzattribut Attr1. Also:

Rel-C(L.ID1, K.ID1, K.ID2, J.ID1, Attr1)

Fertig!!!

Nochmal alles zusammengeschrieben:
K(ID1, ID2)
JJ(Attr2, Attr3, J.ID1)
L(ID1, Attr2, K.ID1, K.ID2)
J(Attr1, ID1, K.ID1, K.ID2)
M(Attr1, ID2, J.ID1)
Rel-B(L.ID1, M.ID2, M.J.ID1)
Rel-A(JJ1.ID1, JJ2.ID1)
Rel-C(L.ID1, K.ID1, K.ID2, J.ID1, Attr1)

Bsp 2 a:
Schauen welcher Buchstabe rechts nicht steht=> Schlüssel=> (???schauen ob man noch einen Buchstaben braucht damit man auf alle kommt???)=> dann die Schritte aufschreiben mit denen man durchgeht, sprich mit dem ersten Ausdruck beginnen und schauen wie man durch kommt. Dann gleiches streichen wie z.b. E zu F und F zu E=> reicht eines von beiden anzuschreiben.

Falls es rechts alle Buchstaben gibt schauen, was man als Schlüssel verwendet sprich mit welchen Buchstabend rechts man zu allen Buchstaben kommt. Falls man mit mehreren Buchstaben alles durchgehen kann, ALLE mit Beistrich angeben (und falls links mehrere Buchstaben stehen, den kompletten Ausdruck immer angeben also nicht nur die einzelnen Buchstaben!)

Nicht vergessen das man das was links steht unterstreicht bzw den ganzen Schlüssel im Ergebnis und man kann =>A weglassen, genauso wenn es Schlüssel ist, man es dann am Schluss nicht nochmal angeben muss!

Bsp 2 b:
Schauen was rechts fehlt bzw wenn es viele Ausdrücke sind, probieren mit welchen Buchstaben man starten und vollständig durchkommt. Wenn es mehrere gibt, die einfach mit Beistrichen getrennt ALLE angeben!
TODO: http://www.informatik-forum.at/showthread.php?t=32160
Verlustlosigkeit: schauen welche Buchstaben es in allen R1, R2,… gemeinsam drinnen gibt.Dann mithilfe der Ausdrücken (….=>….) schauen ob man von oben wenn man die von unten gemeinsamen Buchstaben oben verwendet, zu den restlichen fehlenden Buchstaben unten in einen der Ausdrücken (R1, R2,..) kommt und somit R1, R2,… komplett abdeckt…
Abhängigkeitstreue: schauen ob die Ausdrücke von oben unten in die Klammern (R1,R2) alle vollständig rein passen (man kann AD=>AC in AD=>A und AD=>C „splitten"!)
3NF: ????
BCNF: ???

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

Zeichen	Min	Max
U (Prim)	Wert vom Primär Schlüssel	Beide Werte zusammen zählen
U (Sek)	1 (da wenn alle gleich=> mind. 1 Tupel)	Beide Werte zusammen zählen
Nat. Join	0	kleinster Wert von beiden
R.Out.Join	0	kleinster Wert von beiden
σ	0	Wert von Relation
–	Großer Wert – kleinem Wert	Großer Wert
A ∩ A	TupelwertA	TupelwertA
X	TupelwertA*TupelwertB	TupelwertA*TupelwertB

$(R$ Nat. Join$)$ − (R.Out.Join$)$=¿ 0/ 0
Pi: verhält sich wie select distinc => duplikate werden herausgelöscht
ein natural join, der aber keine werte zum joinen finde verhält sich wie das kartesische produkt (x)=> 0/TupelwertA*TupelwertB

Bsp 6 (Quelle: http://www.informatik-forum.at/showthread.php?p=387344):

Also ich mache das so: 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 erwänen ist: Wenn bei der Linksreduktion auf der linken Seite z.B. 3 Buchstaben stehen, dann nicht von jedem einzelnen Buchstaben eine Attributhülle bilden, sondern zuersteinmal 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.

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.

U Vereinigung =>Zusammen nehmen

$(Assistent)$ U $(Professor)$… 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