Analysis of normal forms for anchor-tables

O. Regardt[1], L. Rönnbäck[1], M. Bergholtz[2], P. Johannesson[2], and P. Wohed[2]
1. Affecto Sweden, 2. DSV, SU/KTH, Stockholm, Sweden

Let R be a relation and A, B, …, Z subsets of the attributes of R.

A *super-key* for a relation R is a superset (not a proper one necessarily) of some candidate key for R. I.e. all candidate keys are super-keys but some super-keys are not candidate keys.

R is said to *satisfy* the join-dependency
* {A, B, … Z} iff
every legal value of R (i.e. every tuple of R) is equal to the join (natural join operator is assumed) of its (R that is) projections on A, B, … Z. In other words: R satisfies * {A, B, … Z}iff R can be nonloss-decomposed into the projections R(A), R(B), … R(Z).

A join dependency * {A, B, … Z}on R is *trivial* iff at least one of A, B, …, Z contains all the attributes of R.

A join dependency * {A, B, … Z}on R is *implied by the candidate key(s)* of R iff each of A, B, …, Z is a super-key for R.

A relation R is in 5NF iff every nontrivial join dependency that is satisfied by R is implied by the candidate key(s) of R.

A table is in 6NF iff it satisfies no nontrivial join dependencies at all.

Note the difference between 5NF and 6NF, for 5NF non-trivial join dependencies may be satisfied as long as these dependencies are implied by the candidate key(s), for 6NF no non-trivial join dependencies at all are allowed to be satisfied.

Anchor-, knot- and attribute-tables

An anchor A(C) is a table with one column. The domain for C is ID. The primary key for A is C.

A(<u>C</u>) is clearly in 6NF, no projection other than on the one attribute is possible.

A knot K(S; V) is a table with two columns. The domain of S is ID, and the domain of V is a data type and never null. The primary key for K is S.

K(S,V) is in 3NF (assuming it is indeed in 1NF, it is in 2NF since the key is not composite, it is in 3NF since no transitive dependencies exist, it is in BCNF since we only have one determinant and it is our one candidate key, it is in 4NF since the existence of multi-valued dependencies implies at least three attributes, now for 5NF: the only two projections (other than π(S, V)) possible is π(S) and π(V). Clearly any natural join between these two projections will not restore the relation K and its original tuples, i.e. the decomposition is not lossless. I.e we have no join-dependencies, other than a trivial one, that K satisfies, i.e. we are in 5NF and 6NF as well.

A static attribute Satt(C;D) for an anchor A(C) is a table with two columns. The domain of C is ID, and the domain of D is a data type and never null. Satt:C is a primary key for Satt and a non-null foreign key with respect to A:C.

S_{att}(C,D) is in 3NF (assuming it is indeed in 1NF, it is in 2NF since the key is not composite, it is in 3NF since no transitive dependencies exist, it is in BCNF since we only have one determinant and it is our one candidate key, it is in 4NF since the existence of multi-valued dependencies implies at least three attributes, now for 5NF: the only two projections (other than π(C, D)) possible is π(C) and π(D). Clearly any natural join between these two projections will not restore the relation S_{att} and its original tuples, i.e. the decomposition is not lossless. I.e we have no join-dependencies, other than a trivial one, that S_{att} satisfies, i.e. we are in 5NF and 6NF as well.

A historized attribute Hatt(C;D; T) for an anchor A(C) is a table with three columns. The domain of C is ID, the domain for D is a data type and never null, and the domain for T is a time type. Hatt:C is a non-null foreign key with respect to A:C and (Hatt:C;Hatt:T) is a primary key for Satt.

H_{att}(C,D,T) contains, as indicated above an anchor surrogate key, a data value and a time stamp (one time stamp for a particular anchor and data value is valid until a new, later, timestamp is stored in the database, e.g the time stamps for one and the same anchor are never the same). H_{att}(C,D,T) is in 3NF (assuming it is indeed in 1NF, it is in 2NF since the only non key attribute is functionally dependent on the whole key only (e.g. the value of the D-column represents a particular D-value wrt the C-column in a particular time point (or from a time point until the next time point for the same C-value to be more precise), it is in 3NF since no transitive dependencies exist, it is in BCNF since we only have one determinant and it is our one candidate key, it is in 4NF since no multi-valued dependencies exist (for one thing, had a multivalued dependency existed (actually the MVD:s always come in pairs) for H_{att} column D would have had to be part of the key and this is not the case), now for 5NF: here we have a number of possible projections (other than π(C, D, T)): π(C), π(D), π(T), π(C, D), π(C, T), π(D, T). This gives rise to a number of possible decompositions that in turn gives rise to possible join-dependencies to test in order to see if H_{att} satisfies them or not:

(i) * {{C,D},{C,T}}
(ii) * {{C,D}, {D,T}}

(iii) * {{C,T},{D,T}}
(iv) * {{C,D},{C,T}, {D, T}} (actually the only one interesting for a 5NF-6NF analysis)
(v) * {{C}, {D, T}}
(vi) * {{D}, {C, T}}
(vii) * {{T}, {C, D}} ((v) through (vii) can be omitted, no natural join can ever restore the original relation.

It is easy to prove that if H_{att} does not satisfy any of (i) through (iii).

(i). Let H_{att} be a relation with tuples ("#1, Green, '920321'", "#1, Blue, '930321'"). Decomposeing H_{att} will yield two relations $H_{attprim}$ (C,D) and H_{attbis} (C,T) with tuples ("#1, Green", "#1, Blue") and ("#1, '920321'", "#1, '930321'") respectively. Natural joining $H_{attpprim}$ with H_{attbis} will yield the following tuples: ("#1, Green, '920321'", "#1, Green, '930321'", "#1, Blue, '920321'", "#1, Blue, '930321'"). Two new spourius tuples are introduced, the decomposition was not loss-less, H_{att} does not satisfy (i). Note that is far easier to prove that a decomposition is _not_ loss-less (i.e. just find one counter example) than the opposite.

(ii). Let if H_{att} be a relation with tuples ("#1, Green, '920321'", "#1, Blue, '930321'", "#2, Blue, '830321'"). Decomposing H_{att} will yield two relations $H_{attprim}$ (C,D) and H_{attbis} (D,T) with tuples ("#1, Green", "#1, Blue", "#2, Blue") and ("Green,'920321'", "Blue,'930321'", "Blue,'830321'") respectively. Natural joining $H_{attpprim}$ with H_{attbis} will yield the following tuples: ("#1, Green, '920321'", "#1, Blue, '930321'", "#1, Blue, '830321'", "#2, Blue, '930321'", "#2, Blue, '830321'). Two new spourius tuples are introduced, the decomposition was not loss-less, H_{att} does not satisfy (ii).

(iii). Let if H_{att} be a relation with tuples ("#1, Green, '920321'", "#2, Blue, '920321'"). Decomposing H_{att} will yield two relations $H_{attprim}$ (C,T) and H_{attbis} (D,T) with tuples ("#1, '920321'", "#2, '920321'") and ("Green, '920321'", "Blue,'920321'") respectively. Natural joining $H_{attpprim}$ with H_{attbis} will yield the following tuples (order of attributes not important): ("#1, Green, '920321'", "#1, Blue, '920321'", "#2, Green, '920321'", "#2, Blue, '920321'"). Two new spourius tuples are introduced, the decomposition was not loss-less, H_{att} does not satisfy (iii).

(iv). *(actually the only interesting one for 5NF))* Let if H_{att} be a relation with tuples ("#1, Blue, '920321'", "#1, Blue, '920330'", "#2, Green, '920321'", "#2, Blue, '920322'"). Decomposing H_{att} will yield three relations $H_{attprim}$ (C,D), H_{attbis} (C,T), H_{attris} (D,T), with tuples ("#1, 'Blue'", "#1, Green", "#2, Green", "#2, Blue"), ("#1, '920321'", "#1, '920330'", "#2, '920321'", "#2, '920322'"), and ("Blue, '920321'", "Blue, '920322'", "Green, '920321'", "Green, '920330'") respectively. Natural joining $H_{attpprim}$ with H_{attbis} will yield the following tuples (order of attributes not important): ("#1, Blue, '920321'", "#1, Blue, '920330'", "#1, Green, '920321'", "#1, Green '920330'", "#2, Green, '920321'", "#2, Green, '920322'", "#2, Blue, '920321'", "#1, Blue '920322'"). A number of spourius tuples are introduced, but remember we have a join more to make (with H_{attris} (D,T)), and this join might rid us from the spurious tuples (theoretically).

However, it does not, the result of the last join will yield the following tuples: ("#1, Blue, '920321'", "#1, Green, '920321'", "#1, Green '920330'", "#2, Green, '920321'", "#2, Blue, '920321'", "#1, Blue '920322'"), still two spurious tuples left, the decomposition was not loss-less, H_{att} does not satisfy (iv).

Since H_{att} does not satisfy any non-trivial join join-dependencies H_{att} is in 6NF.

Let K(S; V) be a knot. A knotted static attribute KSatt(C; S) for an anchor A(C) is a table with two columns. The domain of KSatt:C is ID, and the domain of KSatt:S is ID. KSatt:C is a primary key of KSatt and a non-null foreign key with respect to A:C, and KSatt:S is a foreign key with respect to K:S.

KS_{att}(<u>C,</u>S) is in 3NF (assuming it is indeed in 1NF, it is in 2NF since the key is not composite, it is in 3NF since no transitive dependencies exist, it is in BCNF since we only have one determinant and it is our one candidate key, it is in 4NF since the existence of multi-valued dependencies implies at least three attributes, now for 5NF: the only two projections (other than $\pi(C, S)$) possible is $\pi(C)$ and $\pi(S)$. Clearly any natural join between these two projections will not restore the relation KS_{att} and its original tuples, i.e. the decomposition is not lossless. I.e we have no join-dependencies, other than a trivial one, that KS_{att} satisfies, i.e. we are in 5NF and 6NF as well.

Let K(S; V) be a knot. A knotted historized attribute KHatt(C; S; T) for an anchor A(C) is a table with three columns. The domain of KHatt:C is ID, the domain of KHatt:S is ID, and the domain for T is a time type. KHatt:C is a non-null foreign key with respect to A:C, and KHatt:S is a foreign key with respect to K:S. (KHatt:C;KHatt:T) is a primary key for KHatt.

KH_{att}(<u>C,</u>S,<u>T</u>) contains, as indicated above, an anchor surrogate key, a foreign key towards a knot-table (i.e a reference to a data value), a time stamp (one time stamp for a particular anchor and (reference to) data value is valid until a new, later, timestamp is stored in the database, e.g the time stamps for one and the same anchor are never the same). KH_{att}(<u>C,</u>S,<u>T</u>) is in 3NF (assuming it is indeed in 1NF, it is in 2NF since the only non key attribute is functionally dependent on the whole key only (e.g. the value of the S-column represents a particular S-value wrt the C-column in a particular time point (or from a time point until the next time point for the same C-value to be more precise), it is in 3NF since no transitive dependencies exist, it is in BCNF since we only have one determinant and it is our one candidate key (here it might be worthwhile to actually prove this but I'll skip it for now, it is not hard to do), it is in 4NF since no multi-valued dependencies exist (for one thing, had a multivalued dependency existed (actually the MVD:s always come in pairs) for KH_{att} column S would have had to be part of the key and this is not the case), now for 5NF: here we have a number of possible projections (other than $\pi(C, S, T)$): $\pi(C)$, $\pi(S)$, $\pi(T)$, $\pi(C, S)$, $\pi(C, T)$, $\pi(S, T)$. This gives rise to a number of possible decompositions that in turn gives rise to possible join-dependencies to test in order to see if KH_{att} satisfies them or not:

(i) * {{C,S},{C,T}}
(ii) * {{C,S}, {S,T}}
(iii) * {{C,T},{S,T}}
(iv) * {{C,S},{C,T}, {S, T}} (actually the only one interesting to analyze for 5NF - 6NF)
(v) * {{C}, {S, T}}
(vi) * {{S}, {C, T}}
(vii) * {{T}, {C, S}} ((v) through (vii) can be omitted, no natural join can ever restore the original relation.

It is easy to prove that KH_{att} does not satisfy any of (i) through (iv) using the same pattern as in the proof of H_{att} .

Since KH_{att} does not satisfy any non-trivial join join-dependencies KH_{att} is in 6NF.

Tie-tables

Tie-tables that are all-key give rise to tables in 6NF (obviously, decomposing an all-key table clearly gives rise to spurious tuples when joining the projections, so no non-trivial join-dependencies will ever be satisfied by an all-key tie relation).

Whether or not non all-key tie-tables are in 6NF depends on what functional dependencies hold in the Universe of Discourse that is to be represented by an anchor model. Non all-key tie-tables are neither more nor less normalized compared to relational tables based on any other modeling approach than anchor modeling. It shall be noted, however, that tie-tables (i.e. tables that correspond to *relationships* in ER-modelling) with n columns and less than n-1 columns in the key is very unusual according to our modeling experiences.