University of Glasgow
Dip / MSc Information Technology
Information Systems and Databases

Tutorial Weeks 9/10 – Normalisation

Richard Cooper

November 26th 2009

1 (a) *Define the term **functional dependency** and then describe the problems that might occur if the left hand side of a functional dependency is **not** the primary key of one of the tables in the database.*

A functional dependency is a relationship between columns of the form C1...Cn -> D1.. Dm in which the values of the D columns are completely determined if you know the values of the C columns. This property emerges from the meaning of the data and not the set of actual data. If the Cs do not form a key then every repetition of the Cs needlessly repeats the Ds.

(b) *Given the following **Universal Relation** and set of **functional dependencies***
U(A, B, C, D, E, F, G, H, I, J, K, L, M)

1	*A,B,C -> D, G*	
2	*A,B -> E,F*	***This means that U is not in 2NF since A, B is a subset of the key***
3	*B, C -> K, L, M*	***This means that U is not in 2NF since B, C is a subset of the key***
4	*B-> H*	***This means that U is not in 2NF since B is a subset of the key***
5	*C-> I, J*	***This means that U is not in 2NF since C is a subset of the key***
6	*E -> F*	***This means that U is not in 3NF since F is dependent on the key only because it is dependent on E which in turn is dependent on the key***
7	*I -> J*	***This means that U is not in 3NF since J is dependent on the key only because it is dependent on I which in turn is dependent on the key***
8	*M -> L*	***This means that U is not in 3NF since L is dependent on the key only because it is dependent on M which in turn is dependent on the key***
9	*L -> K*	***This means that U is not in 3NF since K is dependent on the key only because it is dependent on L which in turn is dependent on the key***

*Create a **second normal form** version of the database.*

R1(A, B, C, D, G)	***This the residue after the problematic columns have been split off and this is now in 3NF***
R2(A, B, E, F)	***This solves the problem with the second FD, but this is still not in 3NF because of the sixth FD***
R3(B, C, K, L, M)	***This solves the problem with the third FD, but this is still not in 3NF because of the eighth and ninth FD***
R4(B, H)	***This solves the problem with the fourth FD and this is now in 3NF***
R5(C, I, J)	***This solves the problem with the fifth FD, but this is still not in 3NF because of the seventh FD***

(c) *Create a **third normal form** version of the database.*

R1(A, B, C, D, G)	***unchanged***
R21(A, B, E)	***This the residue of R2 after the problematic column F has been split off and this is now in 3NF***
R22(E, F)	***This solves the problem with the sixth FD and is now in 3NF***
R31(B, C, M)	***This the residue of R3 after the problematic columns K and L have been split off and this is now in 3NF***
R32(L, M)	***This solves the problem with the eighth FD and is now in 3NF***
R33(K, L)	***This solves the problem with the ninth FD and is now in 3NF***

R4(B, H)	***unchanged***
R51(C, I)	***This the residue of R2 after the problematic column J has been split off and this is now in 3NF***
R52(I, J)	***This solves the problem with the seventh FD and is now in 3NF***

(d) *List the **foreign keys** in the resulting third normal form database, listing them in the form: Column A of Table T refers to Column B of Table U.*

R1(A, B) refers to R21(A,B)

R21(E) refers to R22(E)

R1(B,C) refers to R31(B,C)

R31(M) refers to R32(M)

R32(L) refers to R33(L)

R1(B) refers to R4(B)

R1(C) refers to R51(C)

R51(I) refers to R52(I)

(e) *Give the **SQL** and **Relational Algebra** expressions to rebuild the universal relation from the tables generated by steps (b) and (c) of this question.*

```
select A, B, C, D, E, F, G, H, I, J, K, L, M
        from R1, R21, R22, R31, R32, R33, R4, R51, R52
            where R1.A=R21.A and R1.B=R21.B and etc.
```

R2 = join(R21, R22, E = E)

R34 = join(R32, R33, L = L)

R3 = join(R31, R34, M=M)

R5 = join(R51, R52, I = I)

RA = join(R1, R2, A=A and B=B)

RB = join(RA, R3, B=B and C=C)

RC = join(RB, R4, B=B)

U = join(RC, R5, C=C)

2 (a) *What are the functional dependencies involved in the tables generated from: a single entity types and its single and multi-valued attributes?*

Example: an Entity Type E with key attribute K, single valued attributes A1, A2 and A3 and multi-valued attributes M1 and M2.

The Universal Relation would have the FD:

K -> A1, A2, A3

but not K -> M1, M2 since there are several values associated with each value of K, so the key of the Universal Relation must be K, M1, M2, i.e.

E(K, M1, M2, A1, A2, A3)

This is not in second normal form because of the first FD and so as we generate these tables:

T1(K, A1, A2, A3)

T2(K, M1, M2)

T1 is in 3NF because there are no FDs of the form Ai -> Aj and the T2 is in 3NF because there is only the key.

Note beyond level of course. However, T2 is not well normalised since we now have unnecessary duplication in that the table will contain every possible combination of M1 and M2. Consider M1 and M2 being multiple phone numbers and e-mail addresses. For staff member S1 with phones P1,P2 and e-mails E1,E2, we would have (S1,P1,E1), (S1,P1,E2), (S1,P2,E1) and (S1,P2,E2). This brings into play Fourth Normal Form which essentially requires no "multi-valued dependencies" – see a textbook for more, but T2 is not in Fourth Normal Form although it is in Boyce-Codd Normal Form. To get it into Fourth Normal you must split it into:

T2(K, M1)

T2(K, M2)

two entity types connected by a binary relationship with each combination of cardinality and participation constraints?

Example: an Entity Types E1 and E2 with key attributes K1 and K2 connected by relationships R which has attribute RA1 and RA2. (Other entity attributes ignored). The Universal Relation would always have the FDs:

FD1 K1 -> other E1 attributes

FD2 K2 -> other E2 attributes

and FD3 K1, K2 -> RA1, RA2

If it's N-1, it would also have:

K1 -> K2 and other E2 attributes and RA1, RA2

and so the UR has K1 as its key and so is:

R(K1, K2, other E1 and E2 attributes, RA1, RA2)

As it has only one key column it is 2NF, but FD2 means it's not in 3NF because we have K1->K2 -> other E2 attributes so we split it into these tables:

T1(K1, other E1 attributes, FK2 references T2(K2), RA1, RA2)

T2(K2, other E2 attributes)

we have every attribute immediately dependent on the sole primary key – hence 3NF

If it's 1-N, it would also have:

K2 -> K1 and other E1 attributes and RA1, RA2

so as we generate these tables:

T3(K1, other E1 attributes)

T4(K2, other E2 attributes, FK1 references T3(K1), RA1, RA2)

we have every attribute immediately dependent on the sole primary key – hence 3NF

If it's 1-1, it would also have:

K1 -> K2 and other E2 attributes and RA1, RA2

and K2 -> K1 and other E1 attributes and RA1, RA2

so we have some choice here, we could follow either of the schemes above and still be in 3NF. Note that if it is partial in either direction, there is still a functional dependency, since given a value of the partial side, you know either that it is null or that it is a particular value of the other side. However, we use the partial participation to choose one or the other to cut down redundancy.

If it is 1-1 and total on both sides, we have the added possibility of a single table with K1 and K2 as candidate keys. Whichever you choose, all the columns are dependent on the key and you have 3NF.

If it's M-N there are no other FDs so as we generate these tables:

T5(K1, other E1 attributes)

T6(K2, other E2 attributes)

T7(FK1 references T5(K1), FK2 references T6(K2), RA1, RA2)

we again have 3NF since all of the columns depend on all of the keys in each table.

a weak entity type connected by an identifying relationship with a strong entity type?

Example a Strong Entity Type E with key attribute SK and single valued attributes A1, A2 and A3 and a Weak Entity Type W with partial key attribute WK and single valued attributes A4, A5 and A6.

The Universal Relation would have the FDs:

SK -> A1, A2, A3

and SK, WK -> A4, A5, A6

The key of the Universal relation must be SK and WK, but it is therefore not in 2NF as columns A1, A2 and A3 only depend on part of the key. To be in 2NF we must split it into:

T8(SK, A1, A2, A3)

T9(SK, WK, A4, A5, A6)

This is in 3NF since all of the columns depend on all of the keys in each table and there are no transitive dependencies.

(b) Why therefore, is a Relational Schema generated from an ER diagram in Third Normal Form?

Because the tables created from entities have only one key and there are no dependencies between the other attributes; the tables created by multi-valued attributes have no further attributes to create transitive dependencies and the tables created for relationships are dependent on the keys of one of one or other entity (or both in the case of M-N relationships) and there are no other dependencies.

(c) The process of normalisation is intended to improve the design of a relational database. Give two measures of database quality that normalisation is intended to improve.

Reduction of redundancy and null values, no spurious tuples created. Therefore, a change to the data might cause a change to the dependencies derived solely from teh current data set.

(d) Why is not possible to determine the functional dependencies from looking at the data in the tables of a database?

The data may be unrepresentative of the constraints imposed by real world semantics.

(e) Why must a relation with only two columns be in Third Normal Form?

If both columns are the key, there is no other column to be dependent on part of the key, i.e. 2NF.

If there is only one key column it must be in 2NF

You need three columns for a transitive dependency so it must also be in 3NF.

(f) For the relation R(A, B, C), give three examples of primary key selection and functional dependencies, one of which means that the table is in Third Normal Forma, one means that the table is only in Second Normal Form and one which means that the table is only in First Normal Form.

3NF: R(A, B, C) A-> B, C

2NF: R(A, B, C) A-> B, B ->C

1NF: R(A, B, C) A-> C