Normalization Continued

In order to be in 3NF a relation must be in 2NF

A relation is 3NF provided for every non-trivial FD $X \rightarrow Y$ that holds on R either X is a super key (say, condition A) or Y is prime attribute (i.e. part of candidate key) (say, condition B).

So, Relation is in 3NF if either A or B is true. Not in 3NF means: NOT(A or B) = A' and B'

A' means X is not a super key. Hence, X is either part of key (prime attribute) or simply non-prime attribute. B' means y is non-prime attribute. Thus, if X is prime and Y is non-prime then $X \rightarrow Y$ denotes prime attribute(s)$\rightarrow$ non-prime attribute. So, partial dependency and violation of 2NF. Thus,, inorder to be in 3NF, relation must be in 2NF.

Note, the other case if X is also non-prime then $X \rightarrow Y$ represents non-prime$\rightarrow$non-prime. That is transitive dependency. Violation of basic property of 3NF.

Thus, 2NF+No transitive dependency lead to 3NF.

Properties of Decomposition

Normalization breaks down complex relation into multiple simple relations. While decomposition takes place it must fulfill the following criterion.

a) **Attribute Preserving:**
   Suppose the universal relation $R=\{A_1, A_2, ..A_n\}$ is decomposed into number of relations $R_1, R_2, \dots, R_m$. Thus, the decomposition $D=\{R_1, R_2, ..R_m\}$. D is attribute preserving if R_1 U R_2 U..U $_m$= R. That is no attribute of universal relation is lost after decomposition.
b) **Dependency Preserving Decomposition**
   Suppose F be the set of FDs that holds on R. Universal relation R has been decomposed into $R_1, R_2, ..R_n$. F^+ be the closure of F. Let F_i be the FDs from F^+ such that FD include attributes only from R_i. It is also known as restriction of F to R_i. Physically, it means F_i can be verified by looking at R_i only. Thus, $F_1, F_2, ..F_n$ are the set of FDs whose satisfaction can be tested by looking only at $R_1, R_2, .. R_n$ respectively. $F' = F_1 \cup F_2 \cup \dots \cup F_n$. For verification of F, no join operation is required.
   Now if $F^+ = F'^+$ then F and F' are equivalent then the decomposition is dependency preserving.
   Satisfaction of F' also implies the satisfaction of F. F' can be verified on the individual decomposed relation and thereby it ensures the preservance of F. Thus, in case of dependency preserving decomposition, no join operation is required to ensure the satisfaction of FDs.

c) **Lossless Decomposition**

Suppose the universal relation R is decomposed into number of relations R_1, R_2, … , R_m with respect to F, a set of FDs. Thus, the decomposition D={ R_1, R_2, ..R_m} is lossless w.r.t. F provided:

Natural join of $\pi_{Ri}(r)$=r . r stands for state of R. Thus $\pi_{Ri}(r)$ forms the state of R_1. So, in other words, if natural join of the state of the decomposed relation results into state of universal relation then it is loss less. Join must not generate any spurious tuples.

Suppose R is decomposed into R_1 and R_2. The decomposition is loss less provided

Either $R_1 \cap R_2 \rightarrow R_1$

Or $R_1 \cap R_2 \rightarrow R_2$

That is they have a common set of attributes which is key of either R_1 or R_2. Note that, if the decomposition gives rise to primary key and foreign key then it is lossless. That was the case for 1NF/2NF/3NF. If the common attributes of decomposed relations are not their PK and FK of them then natural join will give rise to unwanted tuples (spurious tuples) as we have discussed earlier in the context of relational algebra and SQL.

BCNF (Boyce-Codd Normal Form)

It is a stricter version of 3NF. In 3NF for every non trivial FD X→Y that holds on R either X is a super key or Y is prime attribute, Second criteria may allow certain redundancy. In BCNF it is dropped.

Thus, a relation R is in BCNF if for every non trivial FD X→Y that holds on R, X is a super key.

Say, X→Y violates the criteria. Then decompose R. From the original schema R remove Y. Form a new relation with Y. Copy X from original schema to new one. In this schema X is the primary key and in the original schema X is the foreign key referring to new one. Thus, the decomposition results in to primary key foreign key association. Hence it is loss-less.

Suppose R(A,B,C,D) is a relation. AB is the key. AD is another candidate key. Other FD is D → B. It does not violate 3NF as B is Prime. But it is not in BCNF. To put it into BCNF, original one will be R (A,C, D) and new one will be R1(D, B). In R1, D is PK. In R, D is FK referring to R1.

Suppose, a relation satisfies 3NF by virtue of second clause (i.e. in X→Y, Y is a prime attribute). In BCNF then Y will be removed from original schema. Thus it may so happen that the candidate key of which Y (a prime attribute) was a part is now splitted between original and new schema. So verification of the FD involving the said candidate key (AB in the said example) cannot be done by looking at single relation. Hence, BCNF may not be dependency preserving.

For normalization steps will be 1NF, then 2NF and thereafter 3NF. Or, 1NF – 2NF – BCNF. Mind it, after BCNF if one checks for 3NF it will be always true. So, if 3NF is the goal then after 2NF go for it. 3NF and BCNF differ only if second criteria of 3NF is present in the FD set.

BCNF and 3NF both provides loss-less decomposition. 3NF guarantees dependency preserving which may not be true for BCNF as discussed earlier. Both reduce redundancy.

For practical purposes, it is quite common to normalize up to 3NF.

Null Value and Dangling Tuple

We have discussed earlier that if the joining attribute contains null (say FK is null and joining is based on PK and FK of two participating relation) then those tuples with null value (say, FK is null) will not appear in the output of join operation.

Consider the relations: EMPLOYEE (<u>ECODE</u>, ENAME, BASIC, DOJ, DCODE , DNAME) is decomposed into DEPT(<u>DCODE</u>, DNAME) and EMP(<u>ECODE</u>, ENAME, BASIC, DOJ, DCODE)

Say, for certain employee in EMPLOYEE tuples DCODE is not known or unassigned. Thus, after decomposition in EMP table DCODE for corresponding EMPLOYEE tuples will have null value. Those tuples will not appear in the output of join on EMP and DEPT. So we will not get all the tuples as in EMPLOYEE. However, it can be resolved with the help of outer join on EMP and DEPT. That will provide all the tuples as in EMPLOYEE.

Remember, null value also behaves in different manner in case of aggregate functions.

So, it is desirable to design the database in a manner so that the occurrence of null values can be avoided. Effort in this direction may give rise to another issue called dangling tuples.

Suppose, EMP(<u>ECODE</u>, ENAME, BASIC, DOJ, DCODE) is decomposed into two relation EMPNEW(<u>ECODE</u>, ENAME, BASIC, DOJ) and EMP_DEPT(ECODE, DCODE). In EMP_DEPT a tuple is added only when department is assigned to an employee. IN EMP_DEPT, ECODE is FK referring to EMPNEW. Suppose, there are certain employee for whom department is not assigned. So there will be no entry in EMP_DEPT. Those will also not appear in the output of join on EMPNEW and EMP_DEPT. But they would have appeared in case null value for DCODE was allowed in EMP_DEPT. These missing tuples are dangling tuples. Thus, if the decomposition is made too far to avoid null values then certain tuples may be missed in join output and those are the dangling tuples. Note that, in joining EMPNEW and EMP_DEPT, ECODE is acting as PK and FK. In EMP (if we do not decompose) whenever an employee is there his/her details are known even if DCODE is not assigned. So, we will have to store those by allowing null value for DCODE. To avoid null in a rigid manner, we have considered further decomposition and it leads to the problem.

Multi-Valued Dependency (MVD) and 4NF

The situation for MVD may arise as a consequence of 1NF. Suppose a schema is as follows: STUDENT(ROLL, NAME, DOB, {SCODE, SNAME}, {ACTIVITY_ID, ACTIVITY_DETAILS}). A student learns multiple subjects and takes part in multiple activities. Each student has unique ROLL (PK). ROLL-> NAME, DOB SCODE→SNAME ACTIVITY_ID→ACTIVITY_DETAILS are the given FDS.

Try to recollect what we discussed during 1NF. Identify the repeating units. Attributes which are correlated and repeat together should be removed from original relation and put them in a new relation. Also copy the primary key (ROLL in our example) in new relation where it will be FK. Independent repeating units should be placed in separate relation. In our case, SCODE and SNAME form a unit, ACTIVITY_ID and ACTIVITY_DETAILS form the other unit. With our understanding after 1NF the schemas are like,

STUDENT(ROLL, NAME, DOB) SUB(ROLL, SCODE, SNAME) ACTIVITY(ROLL, ACTIVITY_ID, ACTIVITY_DETAILS)

PKS are identified (underlined) going by FDs. 2NF will decompose SUB and ACTIVITY as follows.

SUB_STUDENT(ROLL, SCODE) SUB_MASTER(SCODE, SNAME) ACTIVITY_STUDENT(ROLL, ACTIVITY_ID) ACTIVITY_MASTER(ACTIVITY_ID, ACTIVITY_DETAILS)

No issue arises. But, if we blindly follow that repeating attributes are removed to form new relation and copy the PK (ROLL in our case) then after decomposition the schemas are:

STUDENT(ROLL, NAME, DOB) SUB_ACTIVITY(ROLL, SCODE, ACTIVITY_ID, ACTIVITY_DETAILS)

PKS are identified (underlined) going by FDs. 2NF will decompose SUB_ACTIVITY as follows.

SUB_ACTIVITY_STUDENT(ROLL, SCODE, ACTIVITY_ID) SUB_MASTER(SCODE, SNAME) ACTIVITY_MASTER(ACTIVITY_ID, ACTIVITY_DETAILS)

Now look into SUB_ACTIVITY_STUDENT(ROLL, SCODE, ACTIVITY_ID) . Suppose a student learns two subjects and participates in two activities. As learning subject and participating in activity are independent issues, to store the information for both the subjects, both the activities are to be stored and resulting into four tuples as follows.

ROLL	SCODE	ACTIVITY_ID	
1	S1	A1	← tuple t1
1	S1	A2	← tuple t2
1	S2	A1	← tuple t3
1	S2	A2	← tuple t4

Let us try to understand informally. Given a Roll we can identify multiple subjects and also for a roll we can identify multiply activities. So ROLL multi-determines SCODE and it is denoted as ROLL →→SCODE. Similarly, ROLL →→ACTIVITY_ID.

Say, X, Y and Z stand for ROLL, SCODE and ACTIVITY_ID respectively. Formally MVD can be defined as:

A multi-valued dependency X→→Y specified on relation schema R where X and Y are subset of R, specifies the following constraint on any relation state r(R):

If two tuples t1 and t2 exist such that $t_1[X]=t_2[X]$then there should exist two more tuples t_3 and t_4 in r(R) with following properties:

$t_1[X]=t_2[X]= t_3[X]=t_4[X]$

$t_1[Y]=t_2[Y]$ and $t_3[Y]=t_4[Y]$

$t_1[Z]=t_3[Z]$ and $t_2[Z]=t_4[Z]$ where Z=R-(X U Y) and t_1, t_2, t_3 and t_4 may not be distinct. [Navathe]

No need to memorize. Follow the previous example of SUB_ACTIVITY_STUDENT. For a particular value of X there are multiple values (say, two) of Y and also for Z. No two tuples are identical. For a particular value of X, Y and Z are repeated. In all such tuples value of X is same. Y and Z will have different possible combinations. Among these tuples if value of Y is same then they must differ in terms of Z. These give the properties mentioned above.

If X→→Y holds on R then X→→Z is also hold. This is denoted as X→→Y|Z. X multi-determines Y denotes that given a particular value of X, the set of values of Y determined by this value of X is completely determined by X alone and does not depend on the values of Z.

A MVD X→→Y in relation R is called trivial if either Y is a subset of X or X U Y = R.

A relation schema R is in 4NF w.r.t a set of dependencies F (functional and multi-valued dependencies) if, for every **nontrivial** MVD X→→Y in F, X is a superkey of R.

Say, X→→Y violates the property of 4NF in a schema. To achieve 4NF, remove Y from the original schema and put it in a new schema. Also copy X in the new schema. In the new schema X and Y form composite PK.

Consider, SUB_ACTIVITY_STUDENT(ROLL, SCODE, ACTIVITY_ID) with MVDs: ROLL→→SCODE and ROLL→→ACTIVITY_ID. Both violate the criteria. Consider any one of the two, say first one. So, a new relation will be formed as SUB_STUDENT(ROLL, SCODE) and in the original one has now ROLL, ACTIVITY_ID on which second MVD is a trivial one. Changing the name the original relation becomes ACTIVITY_STUDENT(ROLL, ACTIVITY_ID). Now, find that instead of blindly putting the repeating attributes in a separate relation during 1NF, if we do it judiciously (i.e. group related attributes to form a unit and put it them one relation, independent units are in separate relation) 4NF can be avoided. Hence, 4NF is a consequence of 1NF.