Normalization Example: Step-by-Step Solution

Dr. Yüe “Jeff” Zhang, October 19, 2016

(Example from Gillenson, ***Fundamentals of Database Management Systems***, Wiley, 2005)

1. Closely study the given data:



4 rows

1. The given table is not a relation yet. Why? Please study the textbook and the PPT slides (the slides on “what is a relation”, and the corresponding textbook pages).

0.1 Bring the above table to 1NF (How? Super simple: repeat all the Sales Person fields, so there will be 12 rows rather than the current 4 rows in the above table. Now this table has become a relation.

Tips: examine be4 1NF

1. A close examination can identify that there are following functional dependencies:

1. SalesPersonNumber 🡺 SalesPersonName, CommissionPercentage, YearOfHire, DeptNumber, ManagerName
2. ProductNumber 🡺 ProductName, UnitPrice
3. SalesPersonNumber, ProductNumber 🡺Quantity

The reason we can concludes 3) is that neither SalesPersonNumber nor ProductNumber alone can determine Quantity:

1. Each sales person sells multiple products, and we do not know which product’s quantity we are talking about if we are only given sales person;
2. Each product can be sold by different sales person and thus have various quantities, we do not know which quantity we are talking about if only given product number. For example: Product 19440 was sold by sales persons Baker and Adams, with quantity sold 473 and 2529, respectively. The same situation exists for 24013 (140 and 3071 units, respectively) and 26722 (688 and 734 units, respectively), among others.
3. From the discussion of a and b above, we have concluded that the quantity sold is jointly determined by SalesPersonNumber and ProductNumber.

We also noticed that DepartmentNumber determines ManagerName – the correspondence of 73 and Scott, and 59 and Lopez, helped to confirm that observation. Therefore there is a relationship between DepartmentNumber and ManagerName.

\*\*\*NOTE:

The above thinking is very typical in normalization:

1. You first examine “whether there are fields working together as the composite key”, which is often the case in a big table;
	1. You then examine what non-key field(s) this composite key determines;
2. Next you examine which part of the composite key alone determines what non-key fields – finding partial functional dependency – the violation of 2NF;
3. Next you examine whether some non-key field(s) could determine some other non-key fields - finding transitive functional dependency – the violation of 3NF;

The above steps are the key in normalization.

1. With the above discussion, we can establish functional dependencies as follows:
2. Full or FD

 (2) Partial or PD (since ProductNumber is only a PART of the primary key)



 (3) Partial or PD (4) Transitive or TD

1. Normal forms violations

From the above figure, we can see that the existing relation (“big table”) violates the 2nd and the 3rd normal forms:

(2) is partial func. depend., which violated 2NF;

(3) is partial func. depend., which violated 2NF;

(4) is transitive func. depend., which violated 3NF;

[We said (4) is transitive func. depend., because DepartmentNumber is a non-key field in the current functional dependency diagram, yet it does logically determine ManagerName – one department has one department manager, so determining a department is in equivalent to determining its manager.

1. Normalization
2. 1NF: Done; see the figure above in the current page.
3. 2NF:
	1. Give all the func. depend.s that violate 2NF their own “independence” – make new relations from the “violating func. depend.s”;
	2. Delete the dependent(s) of those “violating func. depend.s” from the original big relation (big table), but keep the determinants in the original relation/table. See below.

They will become the foreign keys

While studying the following figures, be careful not to click inside or near the figures, because I did not make new figures but used the original figure, masking out the parts to be removed with boxes. Your clicking could change the figure and cause confusion.

 (1) FD

 (2) PD removed – the non-key fields in this func. depend. removed



 (3) PD – same treatment as the above (4) TD

[NOTE] In this SPECIFIC example, the fields involved in 3NF violation HAPPENED TO also disappear, because the logical key of that part – DepartmentNumber is one of the fields removed with “Partial FUNC DEP (3)”.

 (2) PD now has its own “independence” – its own table/relation:



 (3) PD also has its own “independence”. Please note that Transitive FD is here (Transitive FUNC DEP will not be removed in this step) (Transitive func dep could be in the “Full FUNC DEP”, or in one of the Partial func deps – it all depends on the logic of the specific question/case)



 (3) Partial func. depend. (PD) (4) Transitive (TD)

Note: at this point PDs are removed but TD is still here

1. 3NF:
	1. Give all the func. depend.s that violate 3NF their own “independence” – make new relations from the “violating func. depend.s”;
	2. Delete the dependent(s) of those “violating func. depend.s” from the original big relation (big table), but keep the determinants in the original relation/table. See below.
	3. – did you find that the 3NF “repair” procedure is actually the same as that for 2NF? You are right: they are the same. The only difference is the nature of the violation (one is the existence of PD, and the other is the existence of TD).

Relational model: I will only show the parts that are changed, which is the last figure in the above, from which the PD (3) with TD (4) now become two separate func. depend.s:

Full func dep (1)

 PD (2) removed – the non-key fields in this func. depend. removed



 PD (2) now has its own “independence” – its own table/relation:





 The old PD (3), now becomes a completely independent relation/table



The old TD (4), now becomes a completely independent relation/table