Lab9-Normalization-Merging

In this lab, you will learn how to:

**Merge sets of 3NF Relations from many user views into one set of 3NF Relations.**

Step 1: Here are the 3NF relations for each user view (from Labs 8 and 9 on the DBS201 site):

1 CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, CustRep ]

2 PART [ PartNo, PartDescr, QtyOnHand, Class, Whse, Price]

**3B is not in 3NF as it violates 2NF. Order date only depends on order number**

3A CUSTOMER [ CustNo, CustName ]

3B CUSTORDER [ CustNo, OrderNo, OrderDate ]

4A ORDER [OrderNo, OrderDate, CustNo ]

4B ORDERDetail [OrderNo, PartNum, NumOrdered ]

4C PART [ PartNum, PartDescr, QuotedPrice]

5A CUSTORDER [ CustNum, OrderNum ]

5B ORDER [ OrderNum, Orderdate ]

5C CUST [CustNum, CustName, RepNo]

5D REP [RepNo, RepName]

**Step 2: Use consistent name for same attribute in each relation**

Example: The attributes PartNo and PartNum represent a Part Number. Select a consistent name such as PartNo. The same for attributes CustRep and RepNo representing a Sales Representative’s Number we can use RepNo. For attributes CustNo and CustNum representing a Customer Number use CustNo. For attributes Price and QuotedPrice representing the Price of a part we will use Price.

1 CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, **RepNo** ]

2 PART [ PartNo, PartDescr, QtyOnHand, Class, Whse, Price]

3A CUSTOMER [ CustNo, CustName ]

3B CUSTORDER [ CustNo, OrderNo, OrderDate ]

4A ORDER [OrderNo, OrderDate, CustNo ]

4B ORDERDetail [OrderNo, **PartNo**, NumOrdered ]

4C PART [ **PartNo**, PartDescr, **Price**]

5A CUSTORDER [ **CustNo, OrderNo** ]

5B ORDER [ **OrderNo**, Orderdate ]

5C CUST [**CustNo**, CustName, RepNo]

5D REP [RepNo, RepName]

**Step 3:** Examine Primary Key of each relation with composite Primary Key to determine if all parts should be part of Primary Key:

3B CUSTORDER [ CustNo, OrderNo, OrderDate ] – CustNo is not required to be part of PK because OrderNo uniquely identifies an order. If Composite Primary Key was left as is then unique values could include (OrderNo 1001, CustNo 2) and (OrderNo 1001, CustNo 5) which is incorrect for application SINCE Customers 2 and 5 would each have an order numbered 1001.

**This would not be the case if it was properly normalized. It may have been left like this in order to get the resulting table 5A below. This not the way to do approach this problem, but we live with it.**

🡪 3B CUSTORDER [OrderNo, CustNo, OrderDate]

5A CUSTORDER [ CustNo, OrderNo ] 🡪 CUSTORDER [**OrderNo, CustNo** ]

**Step 4:** Use consistent name for relations with same attributes in Primary Key:

1 CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo ]

2 PART [ PartNo, PartDescr, QtyOnHand, Class, Whse, Price]

3A CUSTOMER [ CustNo, CustName ]

3B **ORDER** [OrderNo, OrderDate, CustNo ]

4A ORDER [OrderNo, OrderDate, CustNo ]

4B ORDERDetail [OrderNo, PartNo, NumOrdered ]

4C PART [ PartNo, PartDescr, Price]

5A **ORDER** [OrderNo, CustNo ]

5B ORDER [ OrderNo, Orderdate ]

5C **CUSTOMER** [CustNo, CustName, RepNo]

5D REP [RepNo, RepName]

**Step 5:** Create one relation for relations having an identical Primary Key (the PK can be a one-part PK or a concatenated PK(2 or more attributes ) but it must match exactly.

1,3A,5C CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo ]

2, 4C PART [ PartNo, PartDescr, QtyOnHand, Class, Whse, Price]

3B, 4A, 5A, 5B **ORDER** [OrderNo, OrderDate, CustNo ]

4B ORDERDetail [OrderNo, PartNo, NumOrdered ]

5D REP [RepNo, RepName]

**Step 6:** Resolve any new transitive dependencies, if any were created in Step 5 (None were in this example)

An example of this would occur with merging of the following relations: **See below**

CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo ]

CUSTOMER [CustNo, CustName, RepName ]

🡪 CUSTOMER [CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo, RepName ]

which now has the transitive dependency of RepName (really determined by attribute RepNo) and would be resolved as:

CUSTOMER [CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo]

REP [ RepNo, RepName ]

After you have normalized and merged, you are almost complete on this part of the design. The next step is to draw an ERD and look for relationships. If there is a relationship but none in the entity attribute list, then add the appropriate foreign keys. We are not going to do this part.

🡪 PART [ PartNo, PartDescr, QtyOnHand, Class, Whse, Price]

**NOTES: If you look at this one carefully, there is a transitive dependency in the above. Perhaps if you asked, suppose one of the parts is stored in warehouse 4, and we deleted the record for that part, we would lose that we had a warehouse 4. Like the above we had Warehouse Number and Warehouse Name you would see the transitive dependency easier. Going back to the original data WHSE was a number. If there were lots of warehouses used across the country, then you probably need a name. If there were 3 warehouses at the rear of the main property and they were numbered 1, 2 and 3, then a name is not needed. You still need a table with Warehouse number as the key and only attribute. Now it may turn out that when the design gets implemented a decision is made to not have a warehouse table then that is a business decision. Remember back in the notes on design there was a violation of 3NF in the address. That is because Postal Code determines City. A reasonable decision was made not create a separate table.**

Now complete the exercise on the last page of this lab and submit.

**Exercise:**

Using the set of 3NF relations from Part A of DBS201 Lab 10 shown below:

1 CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo ]

2 PART [ PartNo, PartDescr, QtyOnHand, Class, Whse, Price]

3 ORDER [OrderNo, OrderDate, CustNo]

4 ORDERDetail [OrderNo, PartNo, NumOrdered ]

5 REP [RepNo, RepName]

Merge the following set of relations obtained from 2 additional user views into the above set of 3NF relations and write complete list of 3NF Relations.

6 REPLIST [ RepNo, RepName, Address, PhoneNo, RegionNo ]

7 REPLISTCUST [ RepNo, CustNo, SalesTargetAmount ]

8 CUST [ CustNo, CustName, Email, Phone]

9 REPSUMM [ RepNum, RepName, RegionName ]

<span style='font-size:14.0pt;mso-bidi-font-size:12.0pt'><![if !supportEmptyParas]><![endif]><o:p></o:p></span>

**ASIDE: Some people will have other questions. These would properly be resolved in the Analysis stage when you were interviewing, gathering views and researching the system to be designed. Failing that you can go back to the users and determine**

**if Sales Target is really by CUSTOMER or by SALESREP (See table 7). If Sales target is by Sales Rep, then it could be an attribute by Sales Rep. If each territory/region belongs to one Sales Rep and each Customer has only one Sales Rep, then the Target by Sales Rep can be determined by adding up all the Targets of Customers serviced by that Sales Rep. Also that regions sales target can be calculated. The best idea might be to assign a Sales Target Amount to a Customer. The correction is as follows for the above 2.**

1 CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, RepNo ]

2 PART [ PartNo, PartDescr, QtyOnHand, Class, WhseNo, Price]

3 ORDER [OrderNo, OrderDate, CustNo]

4 ORDERDetail [OrderNo, PartNo, NumOrdered ]

5 REP [RepNo, RepName]

6 WAREHOUSE [WhseNo]

7 REPLIST [ RepNo, RepName, Address, PhoneNo, RegionNo ]

8 CUST [ CustNo, CustName, Email, Phone, Sales Target Amount]

9 REPSUMM [ RepNum, RepName, RegionName ]

1 CUSTOMER [ CustNo, CustName, CustStreet, CustCity, CustSt, CustZip, Email, Phone,

Sales Target Amount, RepNo(FK) ]

2 PART [ PartNo, PartDescr, QtyOnHand, Class, Price, WhseNo(FK)]

3 ORDER [ OrderNo, OrderDate, CustNo(FK)]

ORDERDetail [ OrderNo, PartNo, NumOrdered ]

4 WAREHOUSE [ WhseNo ]

5 REPLIST [ RepNo, RepName, Address, PhoneNo, RegionNo, RegionName ]