LẠI TRUNG MINH ĐỨC – SE62220  
Class: IA1161  
Subject: Introduction to Database

**LAB 2: FUNCTIONAL DEPENDENCY AND NORMAL FORM**

1. **Decompose relations into 3NF relations.**

* Invoices (OrderID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductID, ProductDescription, ProductStandardPrice, OrderedQuantity)
* OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress **(1)**
* CustomerID → CustomerName, CustomerAddress **(2)**
* ProductID → ProductDescription, ProductStandardPrice **(3)**
* OrderID, ProductID → OrderedQuantity **(4)**

**Step 1: Convert to 1NF**

To make the relation satisfy 1NF, we need 2 conditions: the first one is the relation have the PK, and the second is every component of every tuple is an atomic value.

To satisfied the first condition, we find the PK of relation R:

* OrderID, ProductID → OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductStandardPrice, OrderedQuantity
* OrderID, ProductID is composite primary keys.

To satisfied the second condition, we just need to assume the relation satisfied (because there is no data here).

**Step 2: Convert to 2NF**

To make the relation satisfy 2NF, we need 2 conditions: A relation that is in 1NF and contains no partial functional dependencies.

We can see that the first condition is satisfied because R is in 1NF after step 1.

To satisfy the second condition, we need to solve partial functional dependencies.

Easy to see that:

* With (1) and (4), there is no need to have OrderID, ProductID to identify OrderDate, CustomerID, CustomerName, CustomerAddress. Just OrderID is enough.
* With (3) and (4), no need OrderID, ProductID to identify ProductDescription, ProductStandardPrice.

Therefore, these are partial functional dependencies.

To solve it, we create new relations that have PK is subset old-PK with non-key attributes fully dependencies to the PK.

* R1 (OrderID, ProductID, OrderedQuantity)
  + S1 = {OrderID, ProductID → OrderedQuantity}🡪 Full dependencies.
* R2 (ProductID, ProductDescription, ProductStandardPrice)
  + S2 = {ProductID → ProductDescription, ProductStandardPrice} 🡪 Full dependencies
* R3 (OrderID, OrderDate, CustomerID, CustomerName, CustomerAddress)
  + S3 (OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress; CustomerID → CustomerName, CustomerAddress} 🡪 Full dependencies

After decompose the R into R1, R2, R3, these relations satisfy 2NF.

**Step 3: Convert into 3NF.**

The relation that satisfy 3NF is: A relation that is in 2NF and in which NO non-key attribute is transitively dependent on the key.

For the first condition, after step 2, our relations satisfy 2NF.

For the second condition, we can see that:

* R1, R2 with S1, S2 satisfy.
* R3 with S3 doesn’t satisfy because S3 has Transitive dependencies.
  + OrderID → CustomerID and CustomerID → CustomerName cause OrderID → CustomerName.
  + And easy to see that CustomerName, CustomerAddress is more depend on CustomerID than OrderID.

To solve it, we create new relations from R3 that separate these attributes which more depend on old-PK and attributes which more depend on other attributes (here is OrderID and CustomerID) and make these special attributes become Key.

* R4 (OrderID, OrderDate, CustomerID)
  + S4 = {OrderID → OrderDate, CustomerID}
* R5 (CustomerID, CustomerName, CustomerAddress).
  + S5 = {CustomerID → CustomerName, CustomerAddress).

***After these steps, we get relations satisfy 3NF:***

* R1 (OrderID, ProductID, OrderedQuantity)
* R2 (ProductID, ProductDescription, ProductStandardPrice)
* R4 (OrderID, OrderDate, CustomerID)
* R5 (CustomerID, CustomerName, CustomerAddress).

*Reference:*

* Hoffer, J.A., Ramesh, V. and Topi, H. (2012) Modern database management. Available at: https://files.pearsoned.de/ps/ext/9780273780038 (Accessed: 26 July 2016).
* Standard query language (no date) Available at: http://rdbms.opengrass.net/2\_Database%20Design/2.2\_Normalisation/2.2.6\_3NF-Transitive%20Dependency.html (Accessed: 26 July 2016).
* Slide 03\_Ch3\_3Slot.pptx (From Slide 76 – Slide 79)

1. **Redesign relations into 3NF relations.**

Relation R(TechNo, TechName, DeptId, Department, JobNo, Date, SchoolID, School, Hoursworked)

**Step 1: Finding FD**

* TechNo → TechName
* TechNo → DeptId
* DeptId → Department
* SchoolID → School
* JobNo → SchoolID
* JobNo → Date
* TechNo, JobNo → Hoursworked

**Step 2: Convert to 1NF**

* Primary Keys: TechNo, JobNo because:
  + TechNo → TechName, DeptID, Department
  + JobNo → SchoolID, School, Date
* TechNo, JobNo → TechName, DeptID, Department, SchoolID, School, Date, Hoursworked.
* The data already satisfy “every component of every tuple is an atomic value”

**Step 3: Convert to 2NF and 3NF**

According to BCNF consequence: “A relation R has only two attributes A and B satisfies BCNF 🡺 R(A, B) is in BCNF”.

Easy to see that if we decompose relation R into small relations by each of its FD, then we have:

* R1 (TechNo, TechName)
  + S1 = {TechNo → TechName}
* R2 (TechNo, DeptId)
  + S2 = { TechNo → DeptId }
* R3 (DeptId, Department)
  + S3 = { DeptId → Department }
* R4 (SchoolID, School)
  + S4 = { SchoolID → School }
* R5 (JobNo, SchoolID)
  + S5 = { JobNo → SchoolID }
* R6 (JobNo, Date)
  + S6 = { JobNo → Date }
* R7 (TechNo, JobNo, Hoursworked)
  + S7 = { TechNo, JobNo → Hoursworked }

From R1 to R7, these relations satisfy BCNF, and will satisfy 3NF too.