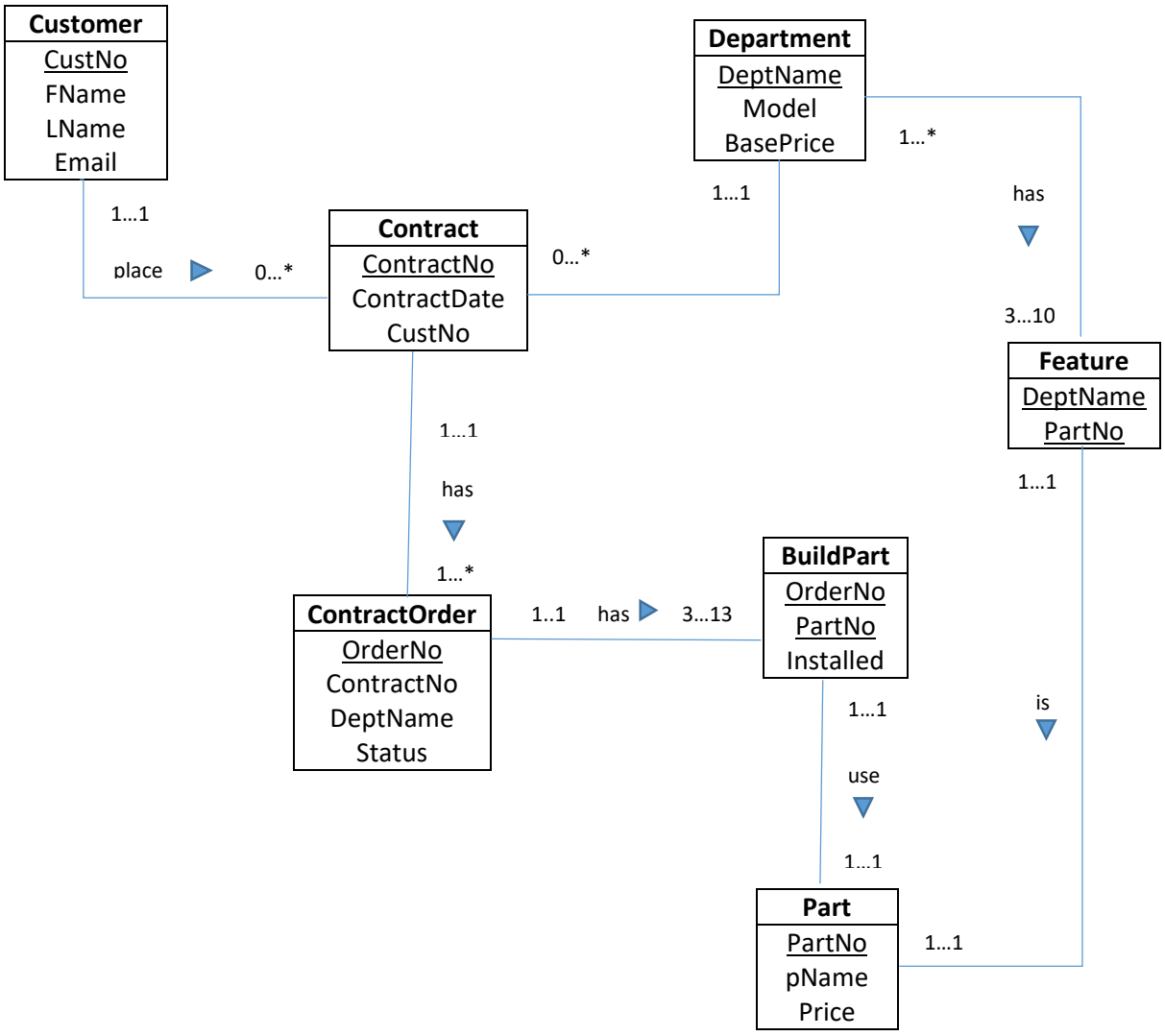


Customer { CustNo, FirstName, LastName, Email }

Department { DeptName, Model, LaborCost }

Part { PartNo, PartName, Price }

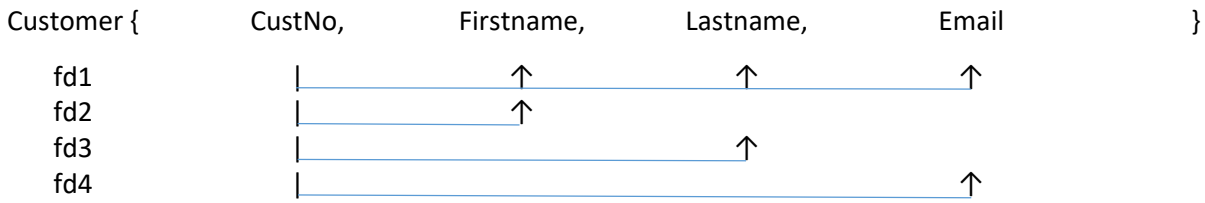
Contract { ContractNo, ContractDate, CustNo }

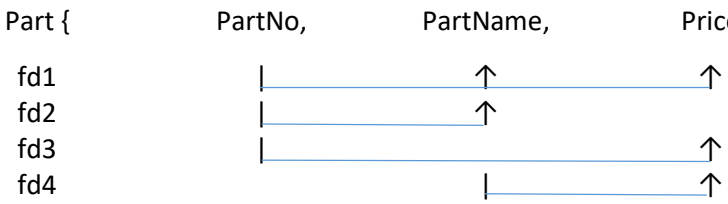
ContractOrder { OrderNo, ContractNo, DeptName, Status }

Build { OrderNo, PartNo, Installed }

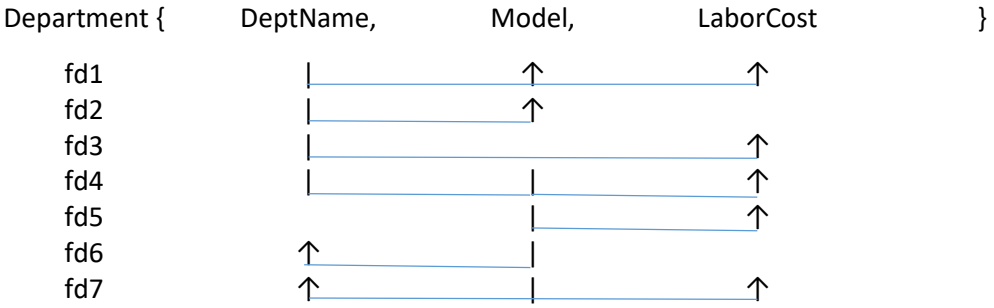
We start our design by first identify the strong entities and weak entities. At first conclusion, relations Customer, Department, and Part relations are strong entities; Contract and Build are weak entities.

Customer and Part are in BCNF because CustNo, and PartNo are superkey for the relations.



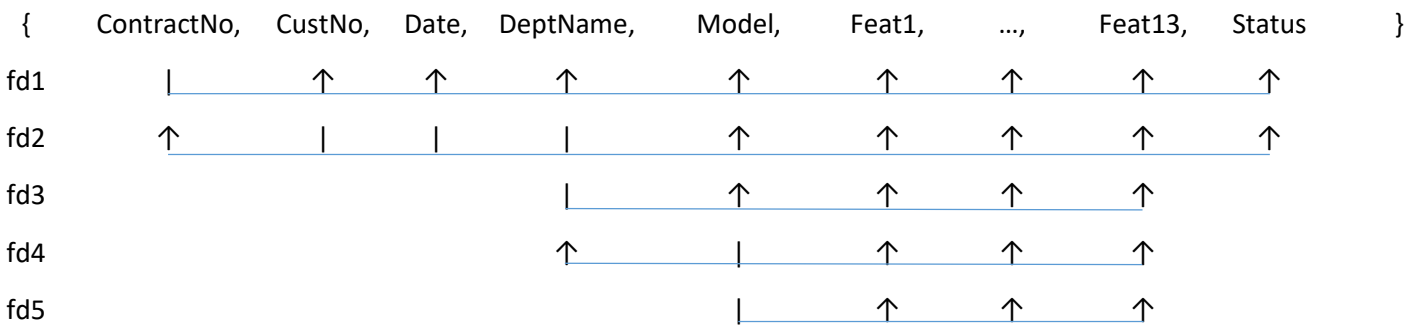


One department is responsible for building one type of ship, therefore the candidate keys for the Department relation are DeptName and Model, because fd2 and fd4. We use DeptName as primary key for Department relation. This relation is in 3NF, because fd2 and fd6 satisfied 3NF condition (b).

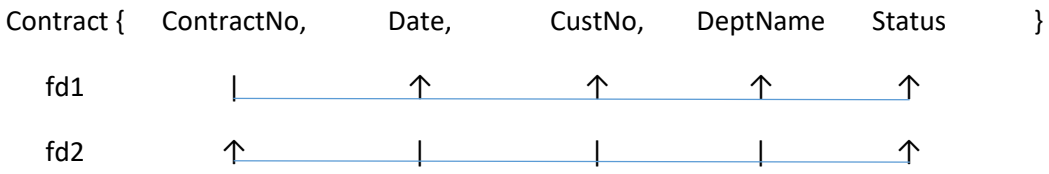


Contract relation is break down result of relation CustomerContractShip. CustomerContractShip is to capture the information needed to process a customer's contract with a department. CustomerContractShip relation shown below is in 1NF, with candidate keys (ContractNO), (CustNo, Date, DeptName), and (ContractNo) choose as primary key for the relation. At first glance, immediately notice that there are many redundant information stored in this relation, decomposition the relation is needed to achieve 2NF.

CustomerContractShip



The CustomerContractShip relation then break down to Contract and Ship. Contract contains two candidate keys, (Contract) and (Date, CustNo, DeptName). Contract is in 2NF because every non-prime attribute is FFD upon every CK of Contract relation; and it is in BCNF because every determinant is a candidate key.

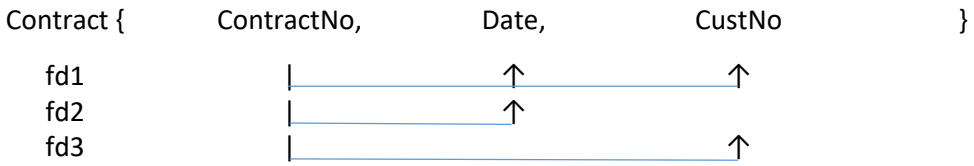


Even though Contract relation is in BCNF, but there exist multi-valued dependency, and they are, Date ->> CustNo, Date ->> DeptName. Whether or not to further decomposed the Contract table, it's all depends on how many contracts are expected to store in the Database in the future, and the business logic of the enterprise.

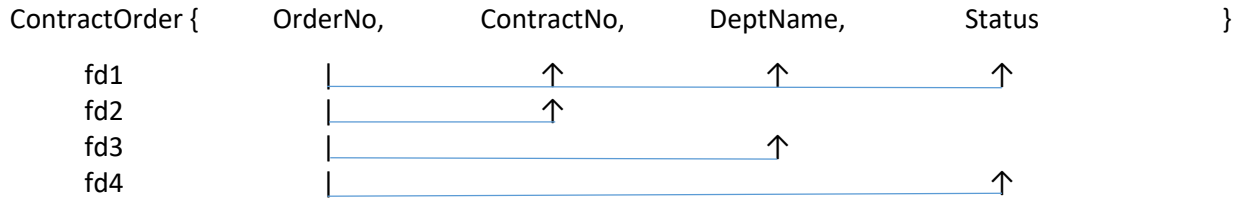
For example, in a particular date, there are N customers, and each customer signed a contract with M different departments, then the value in Date will repeatedly appear N x M times, only one value is needed, and N x M – 1 data are redundant. Redundant data store in CustNo and DeptName are M x N – M and N x M – N, the total of redundant data is 3MN – (M + N + 1). Decomposed the relation into Contract and ContractOrder as shown below, Contract will contains N -1 redundant data, and ContractOrder will have M x N – M and N x M – N redundant data, the total of redundant data is 2MN – (M + N + 1). The redundant data before the decomposition compared with the redundant data after the decomposition is not much of improvement if values of M and N are small; but for large values of M and N, the decomposition of the contract relation will scale nicely.

Break down Contract into Contract and ContractOrder not only help the enterprise store the data more efficiently, also help enterprise manage business logic and its data. Contract relation will store customer contract related information, and ContractOrder will store contract and department related information. Status attribute in ContractOrder indicate the status of the sub-contract. When customer scraped a ship under construction, the sub-contract will also scraped. If all the sub-contracts are terminated, the main contract will also be terminated.

The Contract relation below is in BCNF, because ContractNo is a superkey of this relation.



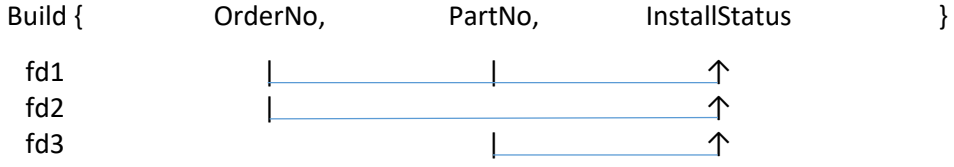
ContractOrder is in BCNF because OrderNo is a superkey of this relation.



Before Build relation is created, Ship relation was a result of the break-down of CustomerContractShip relation, which contained information about the parts a customer ordered to construct the desired ship. Assume each type of ship are consist by a set of common basic part, and a unique set of features. The basic parts are Hull, Engine, and Wing, and feature parts are some parts in the Part table. This table contain multi-valued dependency, to solve this, need to rethink about relation Ship.

Ship { DeptName Feat1 , …, Feat13 }

For each unique ship to be built, a customer must have a contract to place an order, therefore each ship is strictly related with each order. Feat1 to Feat13 are the part number that used to build the ship, together with the order number, the Ship relation is then changed to Build relation. The primary key is {OrderNo, PartNo}, together they uniquely identify each tuple. The foreign keys are OrderNo, and PartNo refereneces to ContractOrder and Part tables. InstallStatus is used to answer queries about the progress of each order/ship. The Build relation is in 3NF, because fd2 and fd3 exist in the relation, therefore 3NF condition (b) is satisfied.



Each model of ship has a unique set of parts, Feature table is used to capture the set of parts that are available for a type of ship. In this relation, because DeptName ->> PartNo, multi-valued dependency constraint exist in the relation, but it is trivial dependency, therefore Feature relation is in 4NF.