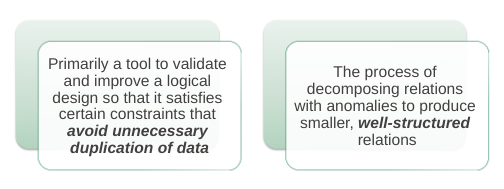
**Database Design & Administration - CST8250**

**Week 3 - January 25th**

Overview

* Normalization can be a dense topic but it will click eventually
* Data normalization defined
* Anomalies
* Normal forms
* Steps to normalization
* Examples

Data Normalization



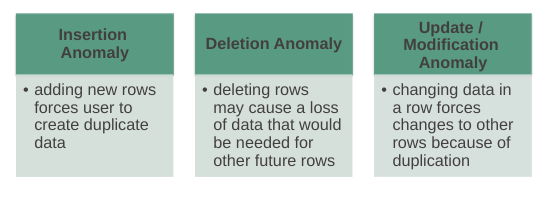
* You want to ideally change data in one spot and not many.
* Relations become tables that are easier to maintain

**Anomalies**

Well-Structured Relations

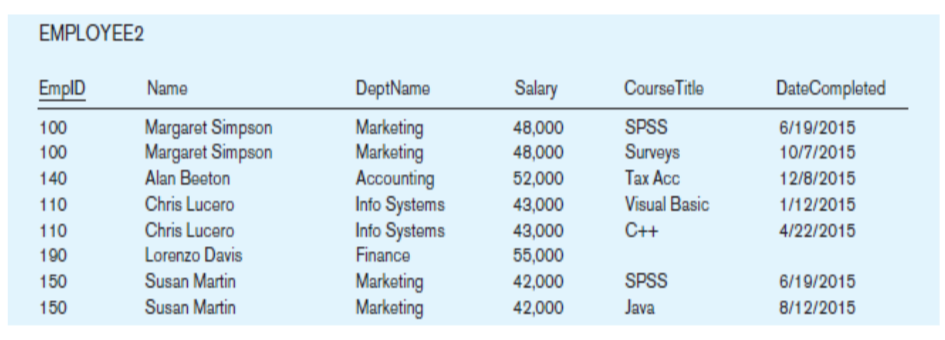
* A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
* Goal is to avoid anomalies
* **Anomaly: problem in un-normalized databases**
* General rule of thumb: A table should not pertain to more than one entity type.

Type of Anomalies



* Insertion anomaly - you are adding something extra you should not have
* Deletion anomaly - you delete something that you cannot recover
* Update/Modification Anomaly - you have to update/modify data in more than one place

Example



—----------------employee—----------------------------------------- —---course—------------------------

* Question–Is this a relation?
  + Answer – Yes: Unique rows and no multivalued attributes
* Question–What’s the primary key?
  + Answer – Composite: EmpID + CourseTitle

Anomalies in this Table

* Insertion – you can’t enter a new employee without having the employee take a class (or at least empty fields of class information)
* Deletion - if we remove employee 140, we lose information about the existence of a Tax Acc class
* Update – giving a salary increase to employee 100 forces us to update multiple records
* Why do these anomalies exist?
  + Because there are two entities (Employee and Course) in this one relation. This results in data duplication and an unnecessary dependency between the entities.

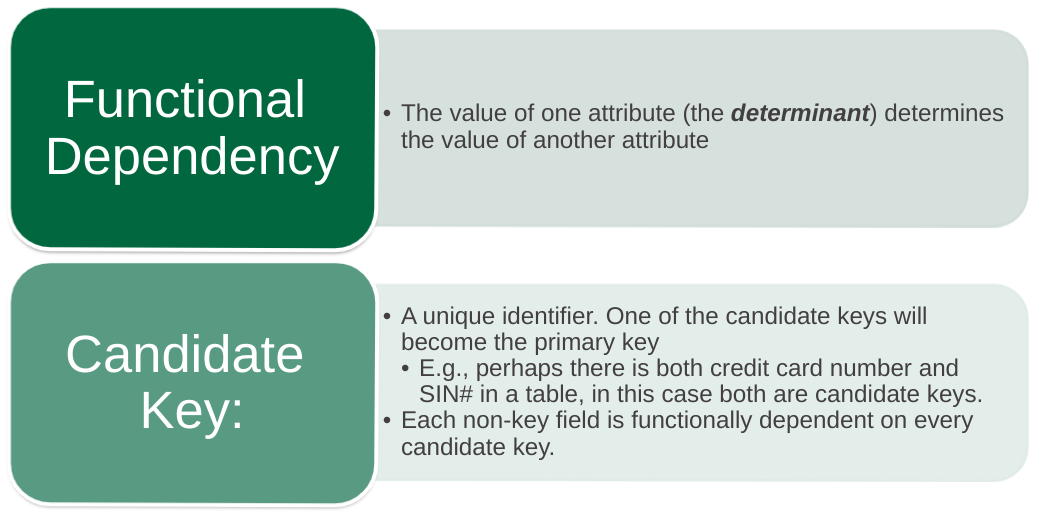
Why should we remove anomalies?

* The main function of a database is to **STORE** and **RETRIEVE** information. We need to be able to do this efficiently and accurately.
* A database that is not well formed is prone to the following issues due to anomalies:
  + Efficiency - increased query time
  + Data integrity – missing, inaccurate, and duplicate information
  + Ease of use - increased effort in writing queries to store and retrieve data

Normal Forms

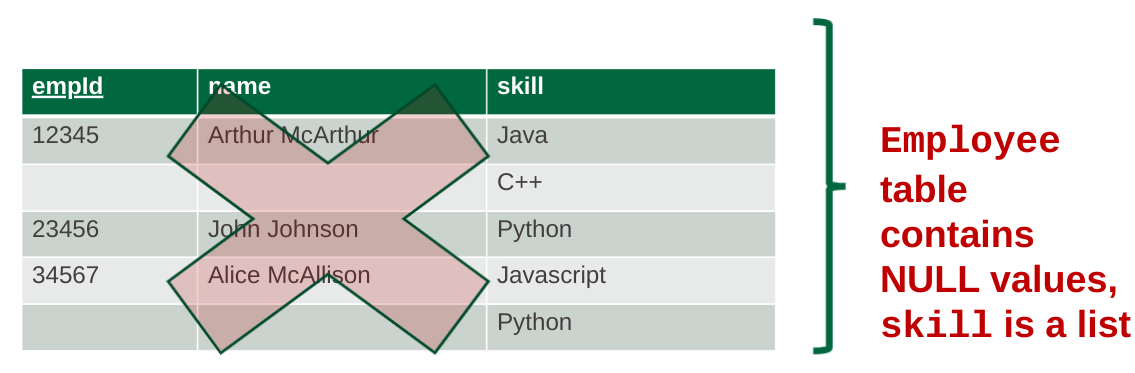
* We want to **REDUCE REDUNDANCY** and **REMOVE ANOMALIES**.
* Our goal is to get to 3rd or Boyce-Codd normal form in order to achieve the above
* We achieve the normal forms by resolving dependencies – our focus is on the keys of the tables
* Recall strong vs weak entities – we will be creating foreign key relationships to resolve these issues

Functional Dependencies and Keys

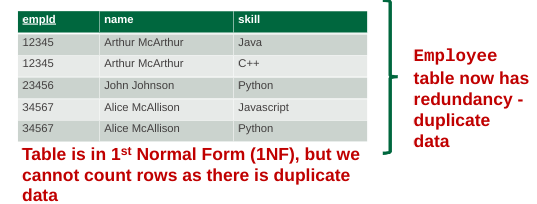


First Normal Form

* **No multivalued attributes**
  + Recall multivalued attributes are lists
  + Storing a list in a database means duplicate data and empty/null entries
* **Every attribute value is atomic**

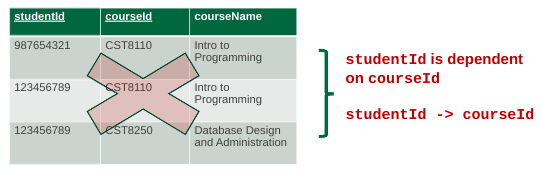


When it turns into the first normal form…

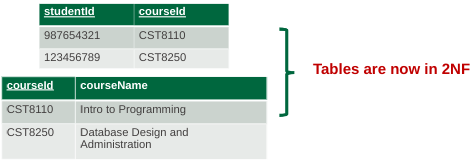


Second Normal Form

* 1NF PLUS **every non-key attribute is fully functionally dependent on the ENTIRE primary key**
  + Every non-key attribute must be defined by the entire key, not by only part of the key
  + No partial functional dependencies
* No multivalued attributes
* **Partial dependencies removed**
  + likely have two entities in one table

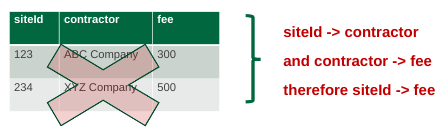


* Courses in only once with the studentID, proper normal form:



Third Normal Form

* 2NF PLUS **no transitive dependencies** (functional dependencies on non-primary-key attributes) (x depends on y, which depends on z, which depends on r etc,)
* Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
* Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as a foreign key in the old table
* **a -> b and b -> c therefore a -> c**
* No multivalued attributes
* **Partial dependencies removed**
* **Transitive dependencies removed**





Boyce-Codd Normal Form (BCNF)

* Relation has more than one candidate key, anomalies may result even though that relation is in 3NF.
* A relation is in **Boyce-Codd normal form (BCNF)** if and only if every determinant in the relation is a candidate key.
* **Simply put, a relation is in BCNF when every attribute/field depends on the key and nothing but the key.**
* Revisit 3NF tables
* Simply put, a relation is in BCNF when every attribute/field depends on the key and nothing but the key.

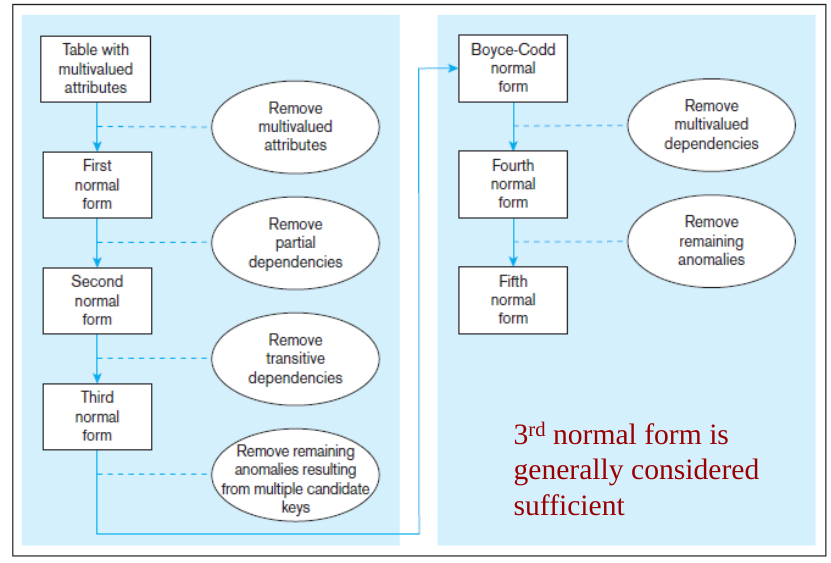


* Revisit 3NF tables
* Simply put, a relation is in BCNF when every attribute/field depends on the key and nothing but the key.

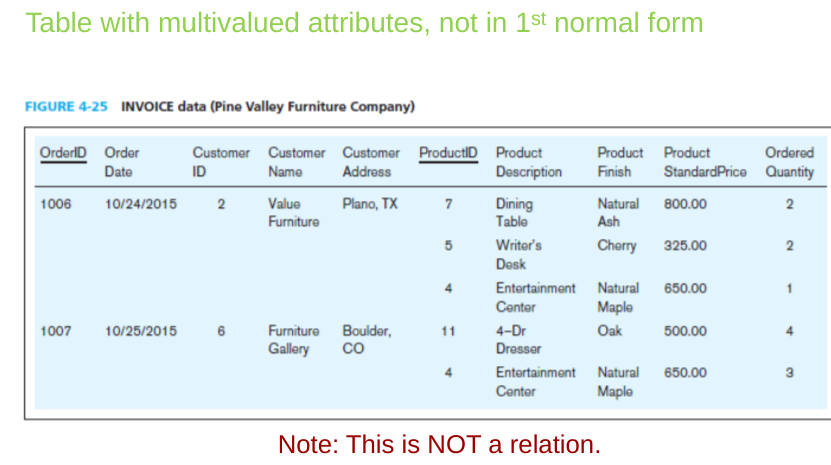


Steps to Normalization Example

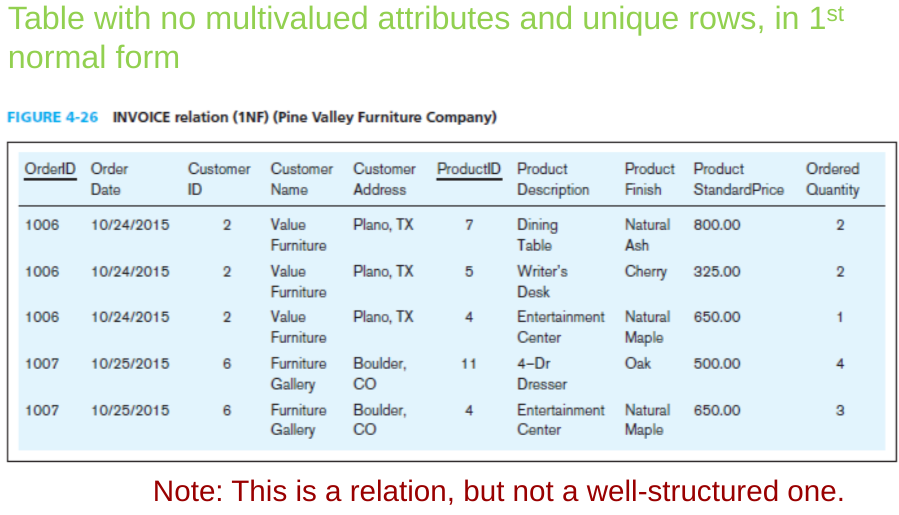
Steps in Normalization

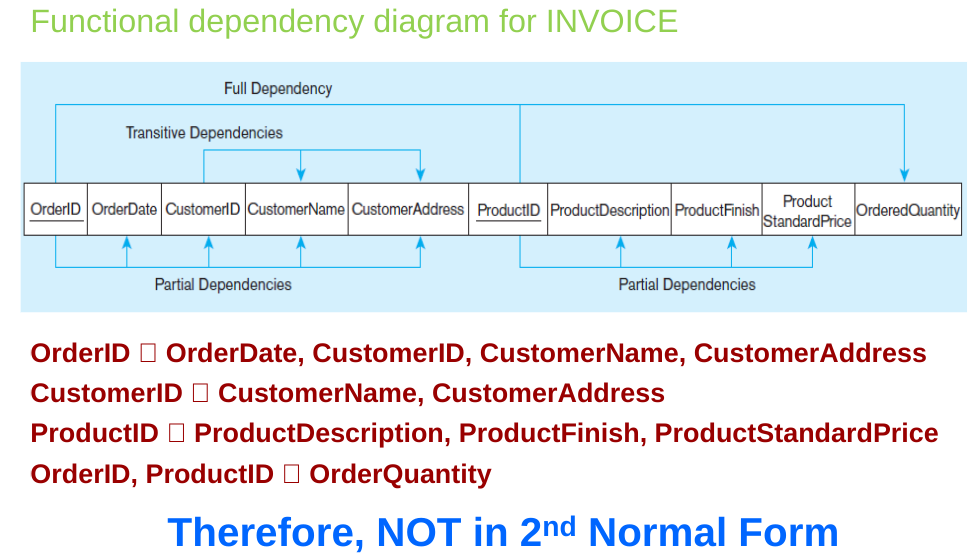


Normalization Example

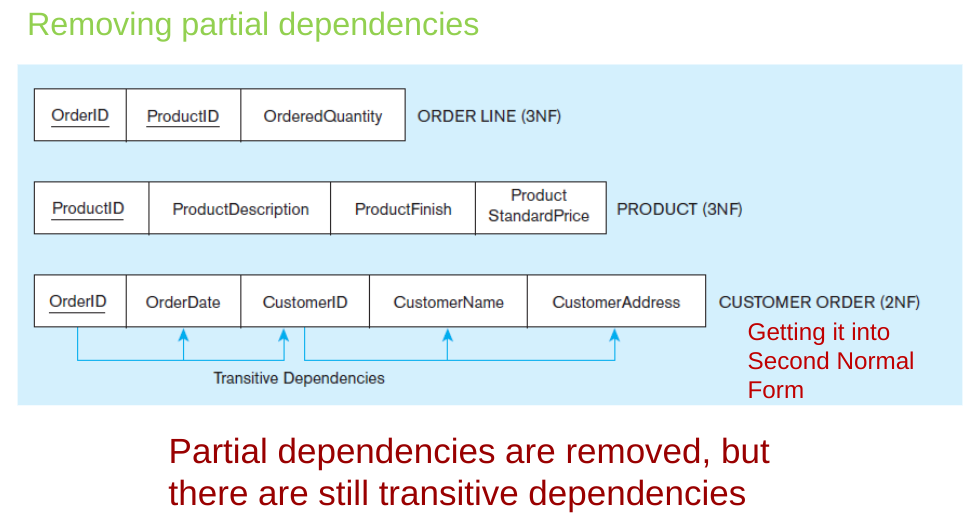


* Orders can contain various amounts of products so it is multi valued.

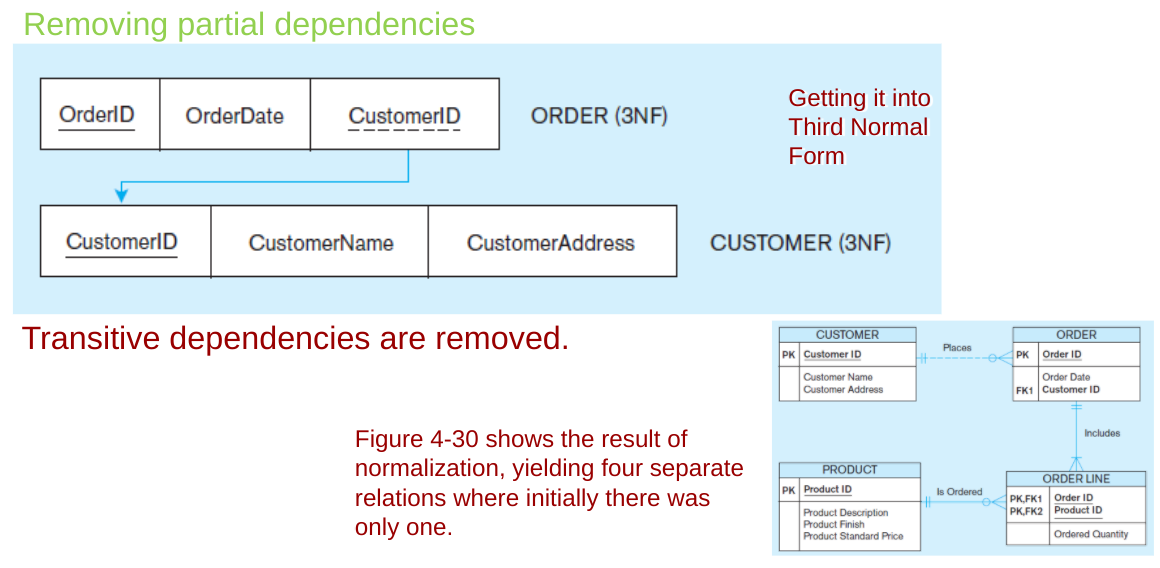




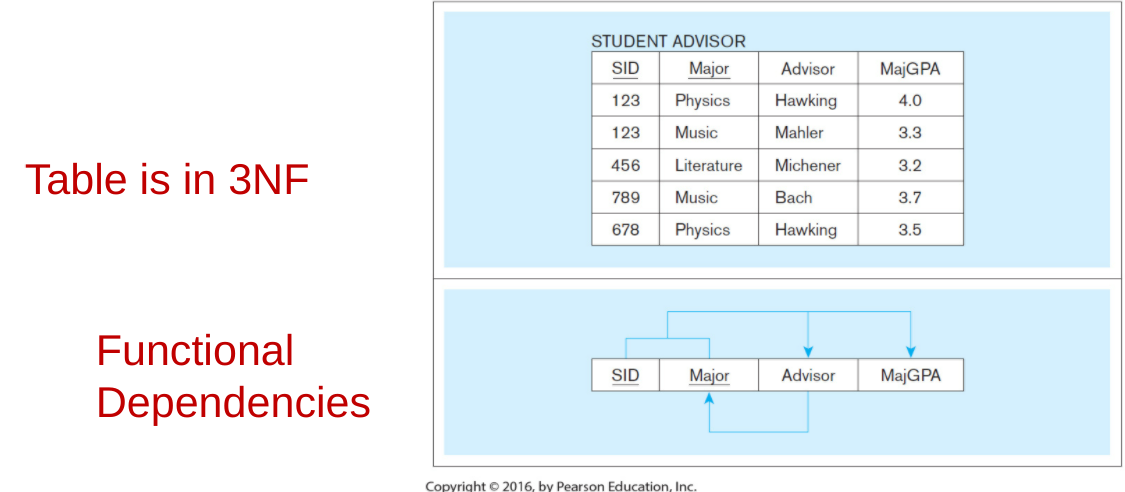
Turn that into…



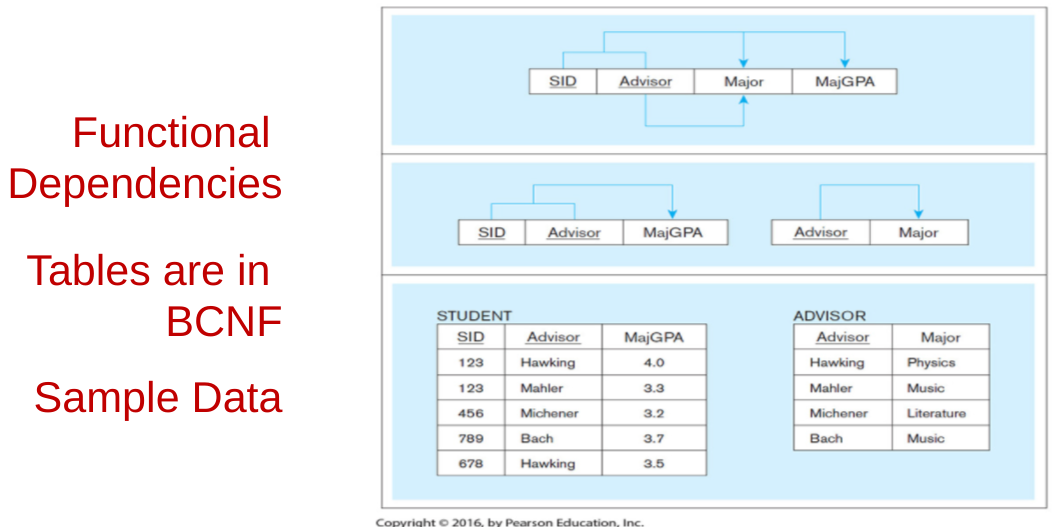
Turn THAT into…



Normalization Example 2: Table in 3NF but not in BCNF



* Arrows indicate dependencies



* There are documents on brightspace titled “Unit 4: Normalization” , “Chapter 3 p 188-208” and “Data Essentials 185” for additional resources on Normalization
* Labs are due soon!!