



Database Design

6-1

Artificial, Composite, and Secondary UIDs



Objectives

This lesson covers the following objectives:

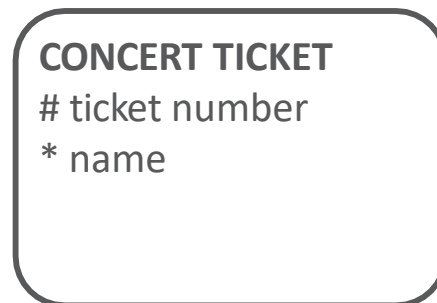
- Define the different types of unique identifiers (UIDs)
- Define a candidate UID and explain why an entity can sometimes have more than one candidate UID
- Analyze business rules and choose the most suitable primary UID from the candidates
- Recognize and discuss the issues of identification in the real world

Purpose

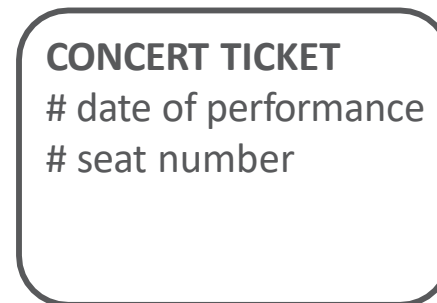
- The unique identifier (UID) is very important in relational databases.
- It is the value or combination of values that enables the user to find that one unique item among all the rest.
- Identifying just the right attribute, or combination of attributes and relationships, is a skill that any database designer must master.
- The unique identifier enables you to find your record in a file, a particular card in a deck of cards, your package in a warehouse, or a specific piece of data in a database.

Simple UUIDs vs. Composite UUIDs

- A UUID that is a single attribute is a simple UUID.
- However, sometimes a single attribute is not enough to uniquely identify an instance of an entity.
- If the UUID is a combination of attributes, it is called a composite UUID.



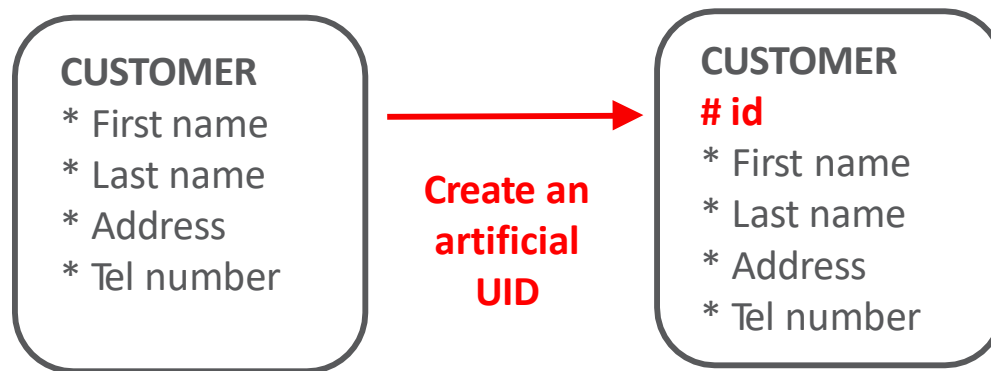
**Simple Unique
Identifier**



**Composite Unique
Identifier**

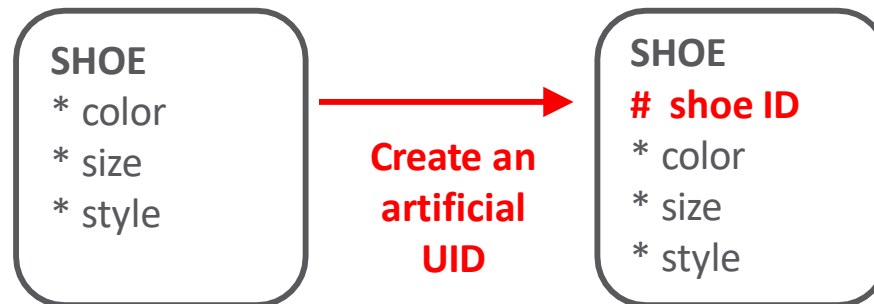
Artificial UUIDs

- Artificial UUIDs are those that don't occur in the natural world but are created for purposes of identification in a system.
- People are not born with “numbers,” but a lot of systems assign unique numbers to identify people: student numbers, customer IDs, etc.



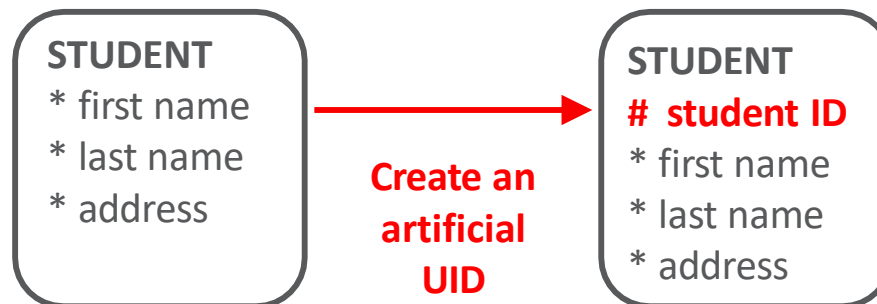
Artificial UUIDs

- A shoe has a color, a size, a style, but no truly descriptive “number.”
- However, a shoe store will assign unique numbers to each pair of shoes so they can be uniquely identified.



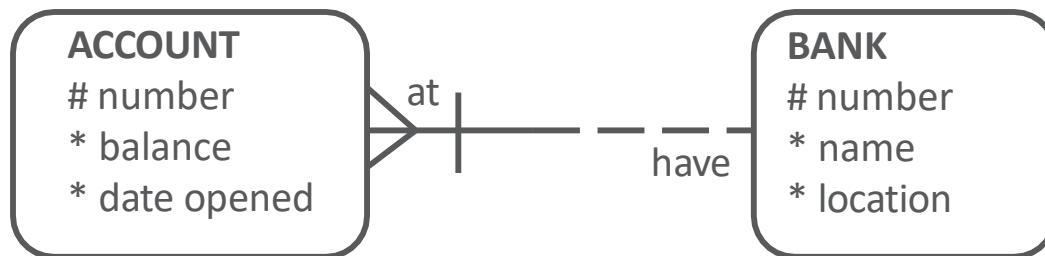
Artificial UID Example

- How can we uniquely identify a STUDENT?
- Could we use a combination of first name and last name?
 - Only if we are sure that the combination is unique.
- Often, it is simpler and more straightforward to create an artificial attribute and make it the unique identifier.
- A UID can be both artificial and composite.



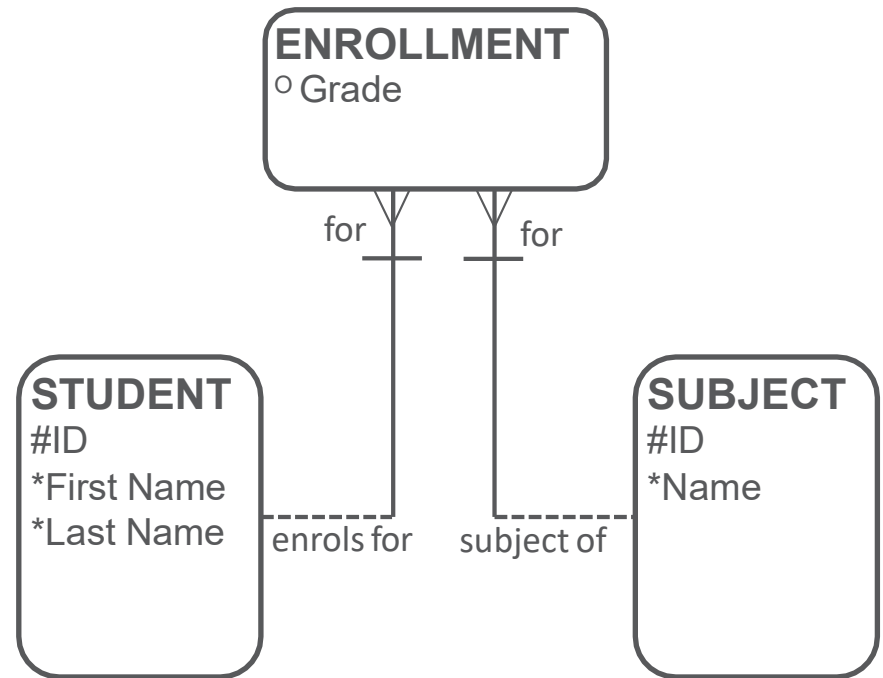
UIDs from Barred Relationships

- Sometimes the UID is a combination of an attribute and a relationship.
- What is the UID of ACCOUNT? Is it artificial? Is it composite?
- Two people could have the same bank account number, but at different banks.
- Bank to bank transfers always need the bank routing number in addition to the bank account number.



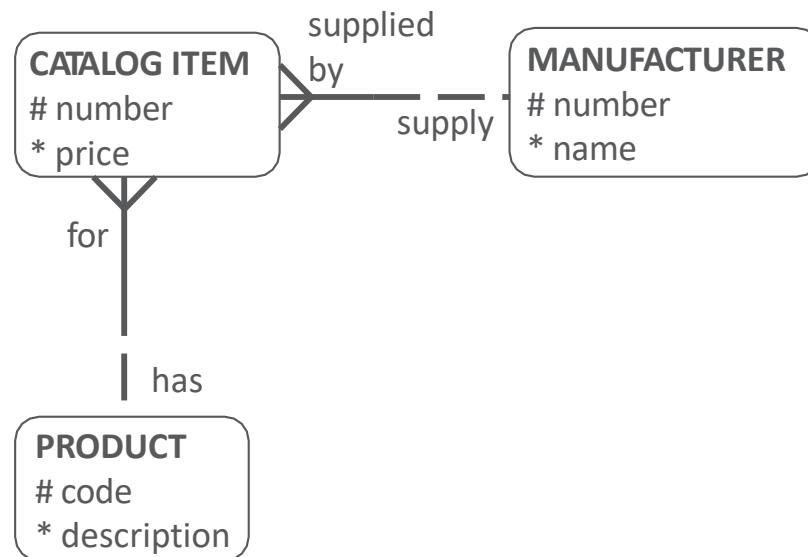
UID from Barred Relationship Intersection Entity

- As we've seen before, the resolution of a M:M relationship often results in barred relationships from the intersection entity to the original ones.
- In this example, the UID of ENROLLMENT comes from STUDENT and SUBJECT.
- The bars on the relationships tell you this.



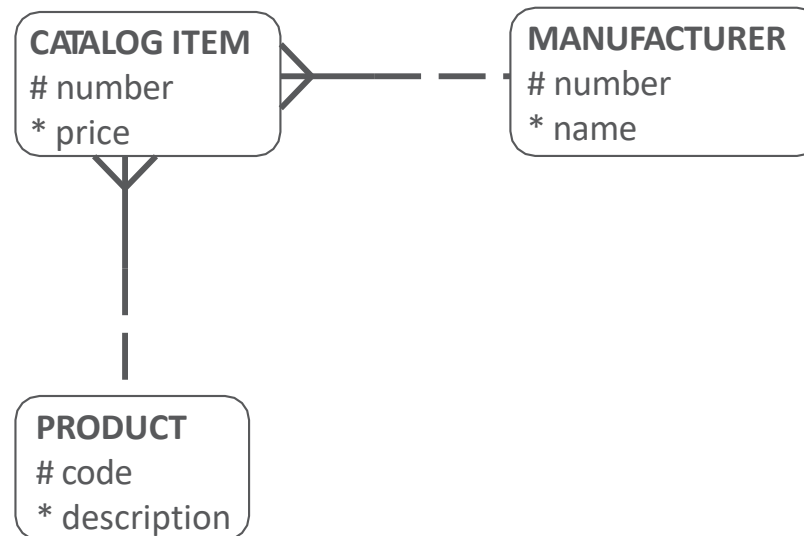
Artificial UID Intersection Entity

- It is possible for an intersection entity to use an artificial attribute as the UID, instead of the barred relationships to the originating entities.



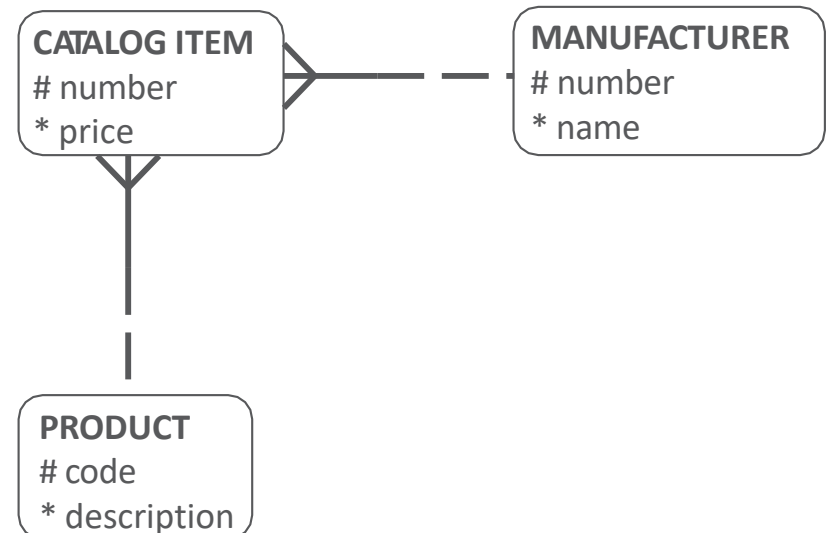
Artificial UID Intersection Entity

- Each MANUFACTURER may produce one or more PRODUCTS (shoes, shirts, jeans, etc.).
- Each PRODUCT may be produced by one or more MANUFACTURERS (Nike shoes, Adidas shoes, Levi's jeans, etc.).



Artificial UID Intersection Entity

- CATALOG ITEM resolves this many-to-many relationship.
- An item in a catalog can be uniquely identified by the manufacturer number and the product code.
- The relationships are not barred, because an artificial UID – catalog number – has been created instead.



Candidate UUIDs

- Sometimes two or more possible UUIDs exist.
- For example, when you order a product from a commercial website, you will usually be assigned a unique customer code and asked to enter your e-mail address.
- Each of these uniquely identifies you, and each could be chosen as the UUID. These are both candidate UUIDs.
- Only one of the candidate UUIDs is chosen as the actual UUID. This is called the primary UUID.
- The other candidates are called secondary UUIDs.

Candidate UUIDs

- Student ID has been chosen as the primary UUID in both of these STUDENT entities.
- The first entity has one secondary UUID, while the second has two secondary UUIDs (one of which is composite).

STUDENT
student ID
(#) badge number
* first name
* last name
* address

One Primary UUID
One Secondary UUID

STUDENT
student ID
(#1) badge number
(#2-1) first name
(#2-2) last name
* address

One Primary UUID
Two Secondary UUIDs

Identification: Database vs. Real World

- Unique identifiers make it possible for us to distinguish one instance of an entity from another.
- As you will see later, these become primary keys in the database.
- A primary key allows you to access a specific record in a database.
- In the real world, however, it is sometimes not so easy to distinguish one thing from another.



Terminology

Key terms used in this lesson included:

- Artificial UID
- Candidate UID
- Composite UID
- Primary UID
- Secondary UID
- Simple UID
- UID

Database Design

6-2

Normalization and First Normal Form



Objectives

This lesson covers the following objectives:

- Define the purpose of normalization in database models
- Define the rule of First Normal Form in the normalization process
- Determine if an entity conforms to the rule of First Normal Form
- Convert an entity to First Normal Form if needed

Purpose

- Think about storing your friends' phone numbers in three different places: your address book, your cell phone, and a sheet of paper that you have taped to your refrigerator.
- It's a lot of work if a friend changes his/her phone number.
- You have to change it in your address book, cell phone, and the sheet of paper taped to you



Purpose

- What happens if data is stored in more than one place in a database?
- What if someone changes the information in one place and not the other—how do you know which information is correct?
- Redundancy like this causes unnecessary problems in a database.



Purpose

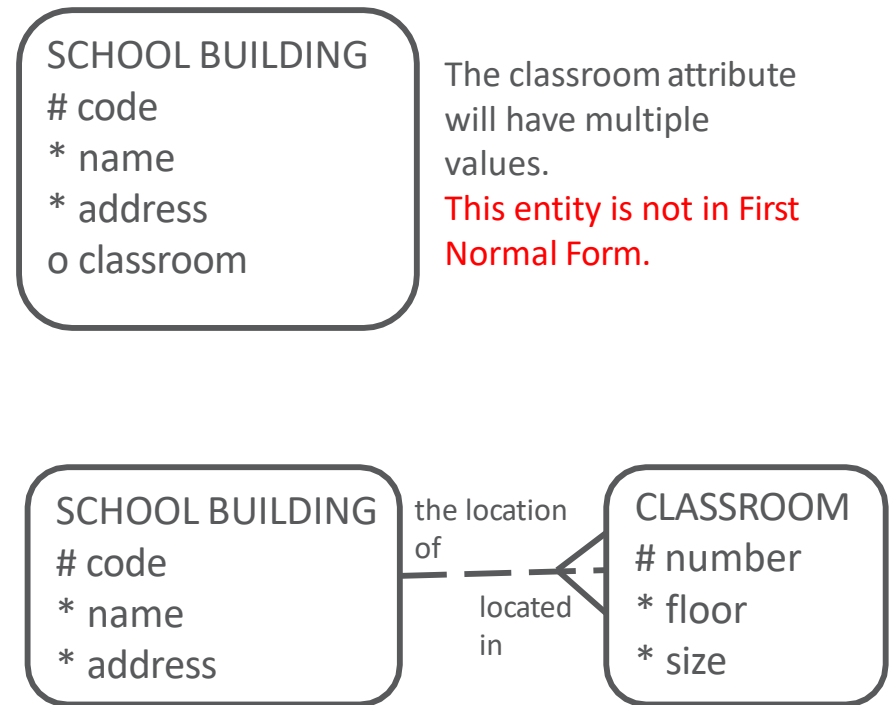
- Normalization is a process that is used to eliminate these kinds of problems.
- One of your goals as a database designer is to "store information in one place and in the best possible place".
- If you follow the rules of normalization, you will achieve this goal.



First Normal Form (1NF)

- First Normal Form requires that no multi-valued attributes exist.
- To check for 1NF, validate that each attribute has a single value for each instance of the entity.
- One code, one name, and one address exist for the school building, but not one classroom.

SCHOOL BUILDING 1NF

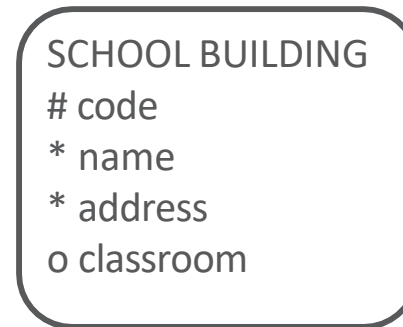


CLASSROOM is now its own entity.
All attributes have only one value per instance.
Both entities are in First Normal Form.

First Normal Form (1NF)

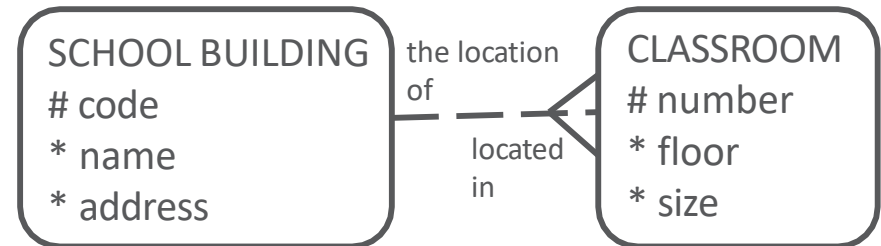
- Since many classrooms exist in a school building, classroom is multi-valued and violates 1NF.
- If an attribute is multi-valued, create an additional entity and relate it to the original entity with a 1:M relationship.

SCHOOL BUILDING 1NF



The classroom attribute will have multiple values.

This entity is not in First Normal Form.

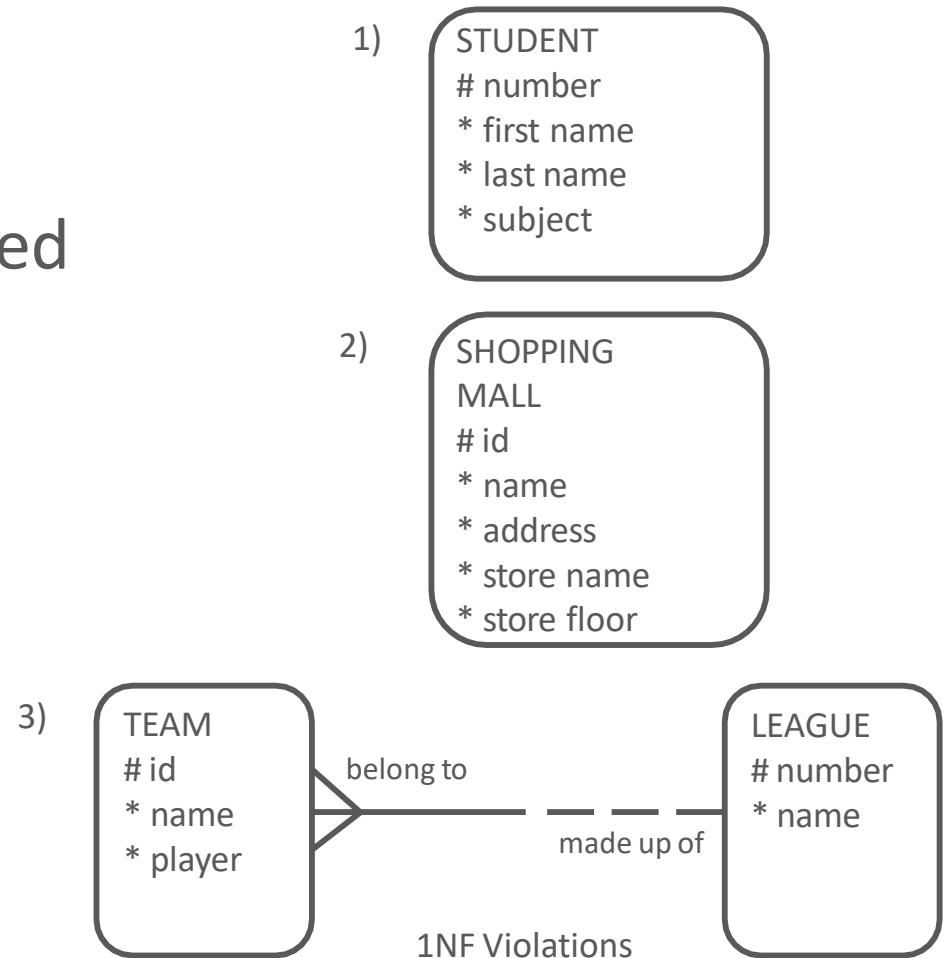


CLASSROOM is now its own entity.
All attributes have only one value per instance.

Both entities are in First Normal Form.

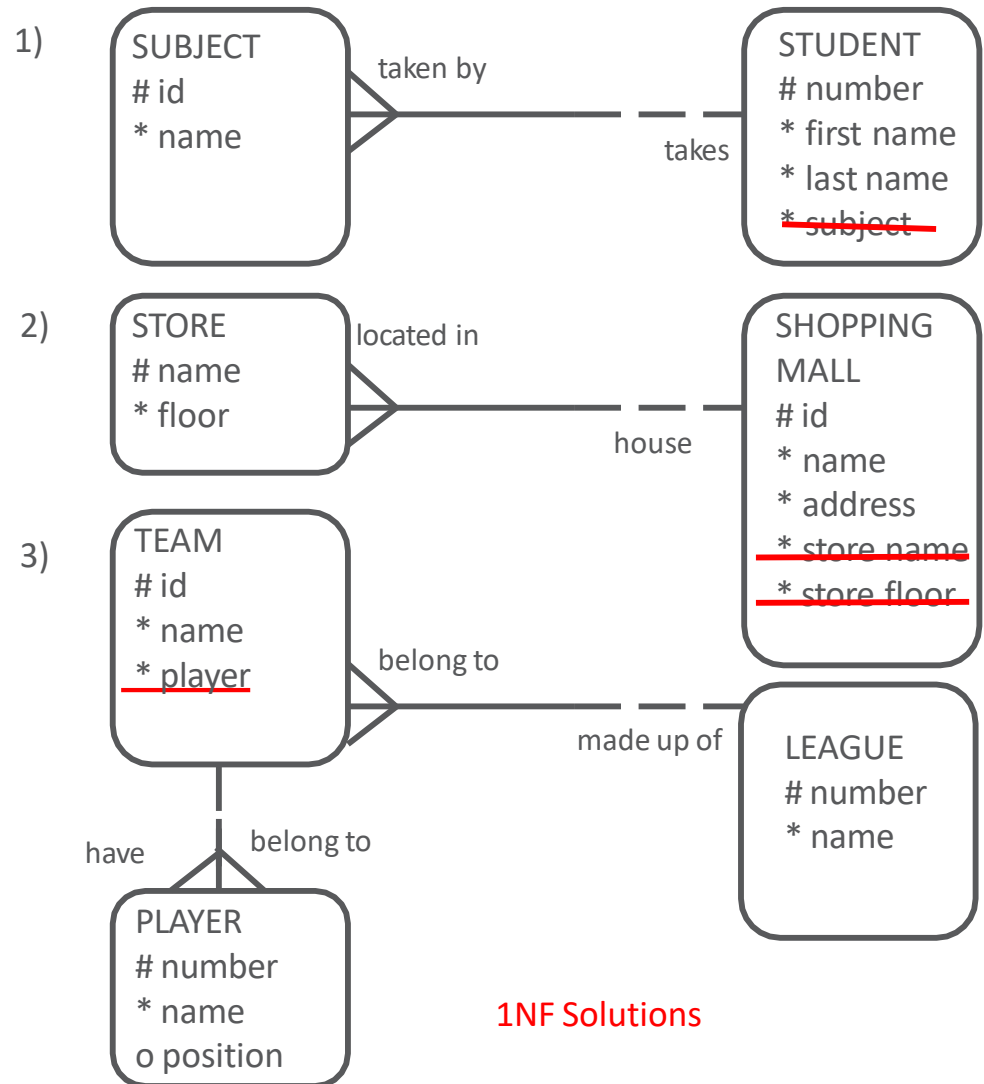
1NF Violations

- Examine the entities.
- Are there any multi-valued attributes?



1NF Solutions

- When all the attributes in an entity are single-valued, that entity is said to be in First Normal Form.



Terminology

Key terms used in this lesson included:

- First Normal Form (1NF)
- Normalization
- Redundancy

Database Design

6-3

Second Normal Form



Objectives

This lesson covers the following objectives:

- Define the rule of Second Normal Form in the normalization process
- Examine a non-normalized entity and determine which rule, or rules of normalization are being violated
- Apply the rule of Second Normal Form to resolve a violation in the model

Purpose

- Your goal as a database designer is to “store information in one place only and in the best possible place.”
- Consistently applying the rules of normalization helps you achieve this goal.
- When organizing information such as your friends’ phone numbers and addresses, you want to make sure that you store that information in the appropriate place -- such as a personal address book.



Purpose

- If you store a friend's address in your recipe box, for instance, you may not find it until the next time you look up that recipe.
- Normalization is a process that helps eliminate these kinds of problems.



Second Normal Form Example

- Examine the entity PRODUCT SUPPLIER.
- The UID is a composite UID

consisting of the supplier number and the product number.

- If one supplier supplies 5 different products, then 5 different instances are created.
- What happens if the supplier name changes?

PRODUCT SUPPLIER

supplier number
product number
* purchase price
* supplier name

Second Normal Form Example

- The supplier name would then need to be changed in 5 different instances.
- What if some of them were changed, but not others?
- How would users know which name is the correct name?

PRODUCT SUPPLIER

supplier number
product number
* purchase price
* supplier name

Second Normal Form Description

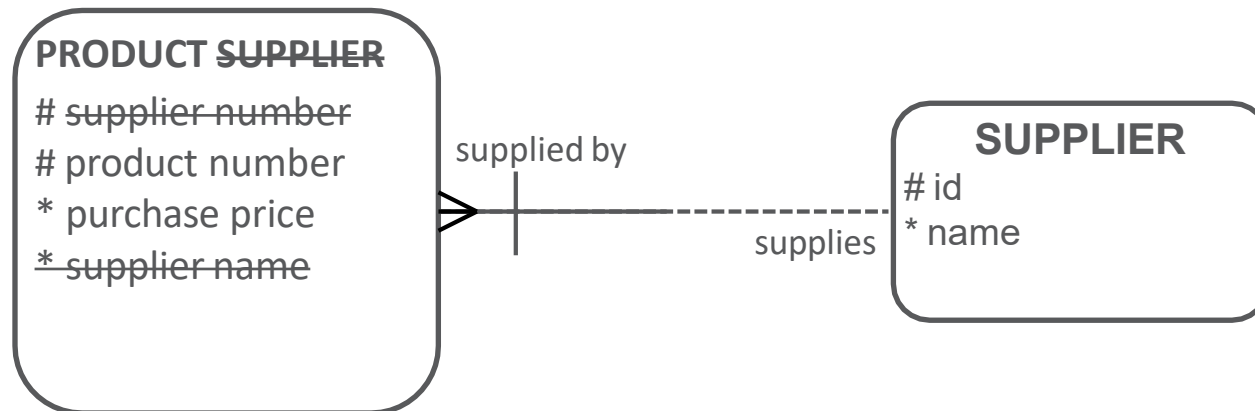
- Second Normal Form (2NF) requires that any non-UID attribute be dependent on (be a property of, or a characteristic of) the entire UID.
- Is purchase price a property of supplier number, product number, or both?

PRODUCT SUPPLIER

supplier number
product number
* purchase price
* supplier name

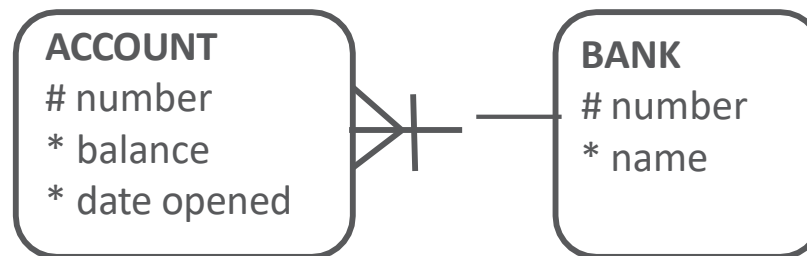
Second Normal Form Description

- Is supplier name a property of supplier number, product number, or both?
- 2NF requires a “both” answer to each question.



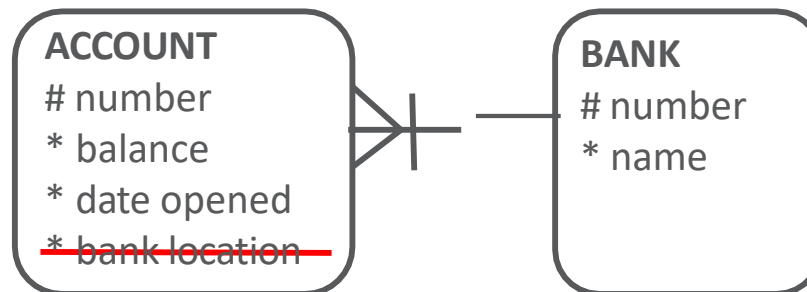
Second Normal Form Bar Relationship

- The UID for ACCOUNT is a composite UID from a barred relationship consisting of ACCOUNT number and BANK number.
- Is balance a property of ACCOUNT number, BANK number, or both?
- Is date opened a property of ACCOUNT number, BANK number, or both?



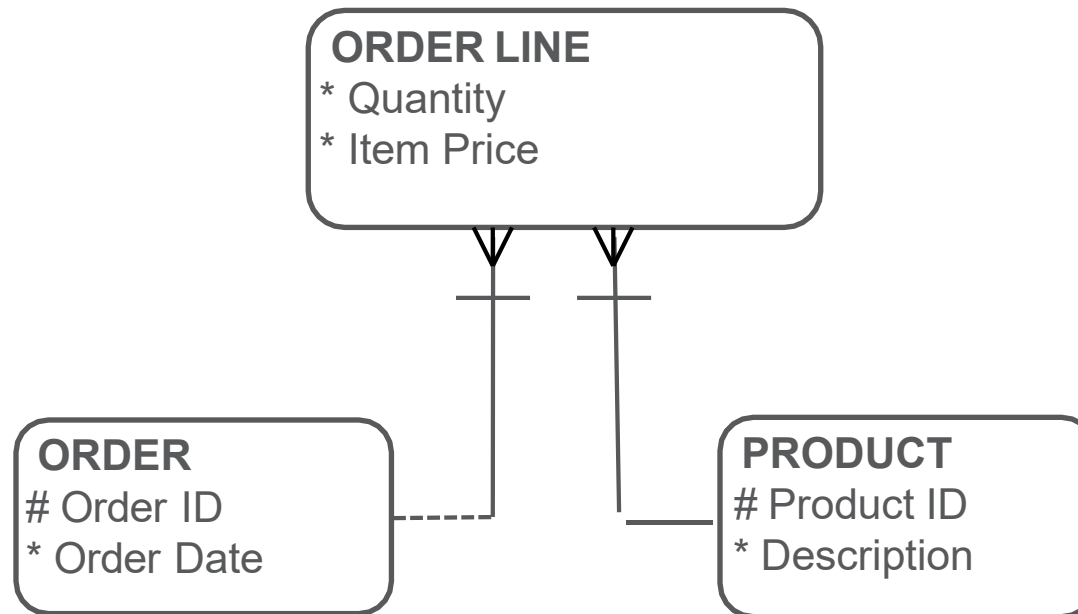
Second Normal Form Violation

- In this ERD, the attribute bank location has been added. Is bank location a property of ACCOUNT number, BANK number, or both?
- It is a property of BANK number only and is thus misplaced. This is a violation of Second Normal Form.
- What would happen if a bank's location changed?
- Every account at that bank would need to be updated.



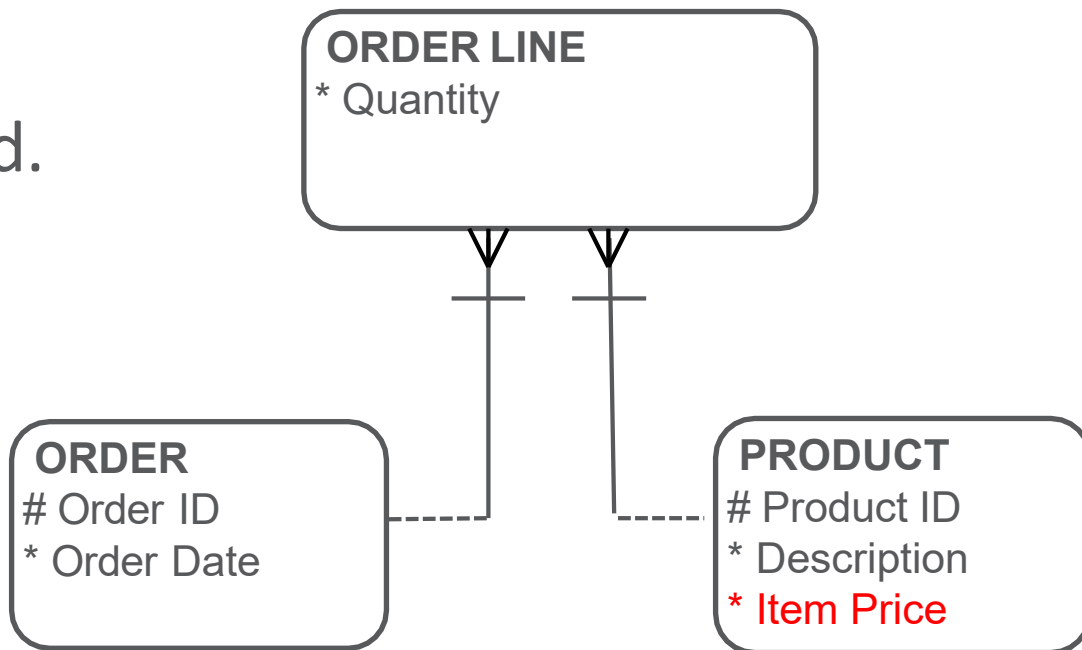
Order ERD

- What is wrong with this diagram?



Order ERD

- The ERD is now in 2NF
- Answer: The price attribute is misplaced.
- Item Price depends solely on PRODUCT.
- This is a violation of Second Normal Form.



Terminology

Key terms used in this lesson included:

- Second Normal Form (2NF)

Database Design

6-4

Third Normal Form



Objectives

This lesson covers the following objectives:

- Identify transitive dependencies in a data model
- Define the rule of Third Normal Form in the normalization process
- Examine a non-normalized entity and determine which rule, or rules of normalization are being violated
- Apply the rule of Third Normal Form to resolve a violation in the model

Purpose

- Your goal as a database designer is to "store information in one place only and in the best possible place."
- Following the rules of normalization helps you achieve this goal.
- You may want to enter different kinds of information for a friend in your personal address book: phone number, address, name of school or place of work.

Purpose

- If you have several friends who go to the same school, and you enter the school's street address along with each of them, you would not only be duplicating data but causing potential problems - for instance, if the school moved and changed its address, you would have to go back and change it everywhere!
- Normalization is a process to eliminate these kinds of problems.



Third Normal Form Rule

- The rule of Third Normal Form (3NF) states that no non-UID attribute can be dependent on another non-UID attribute.
- Third Normal Form prohibits transitive dependencies.
- A transitive dependency exists when any attribute in an entity is dependent on any other non-UID attribute in that entity.

Third Normal
Form Violation

CD
id
* title
* producer
* year
o store name
o store address

Third Normal Form Rule

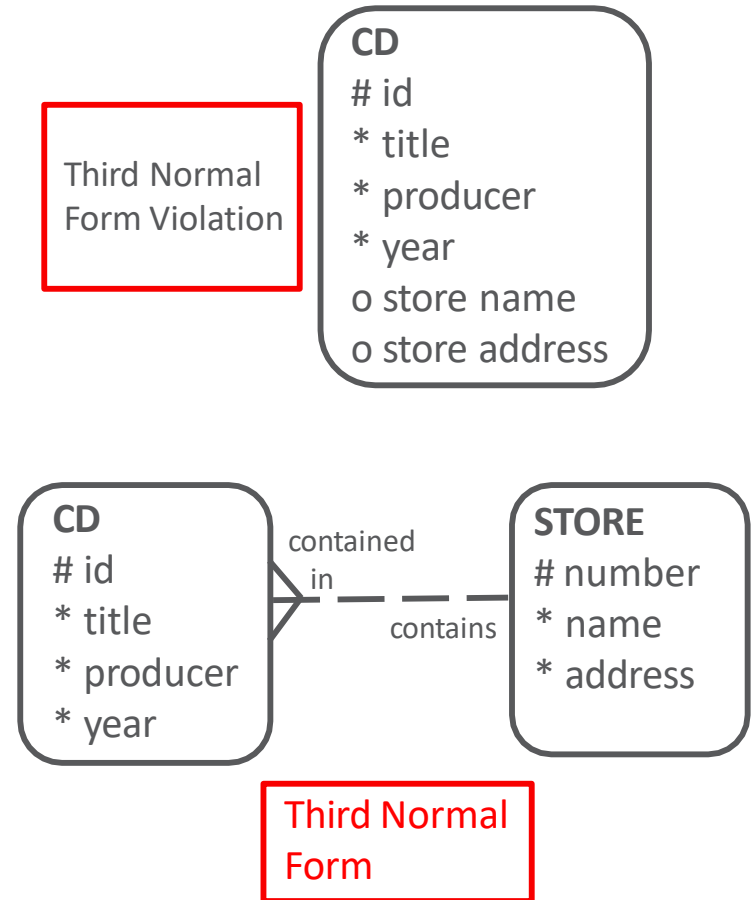
- Think of the kind of information you'd like to store about your CD collection.
- Does information about the store where you bought the CD belong in the same entity?
- If the store address changed, you would have to change the information on all the CDs that were bought at that store.

Third Normal
Form Violation

CD
id
* title
* producer
* year
o store name
o store address

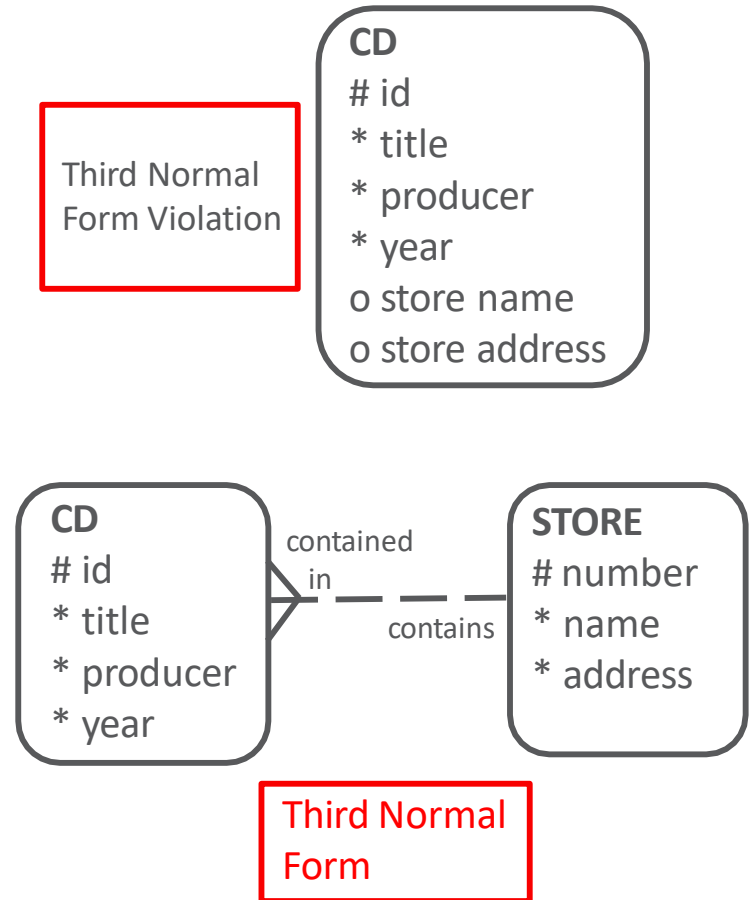
Third Normal Form Transitive Dependency

- The store address is dependent on the CD number, which is the UID of the CD entity. So this entity is in 1NF and 2NF.
- But store address is also dependent on store name, which is a non-UID attribute.
- This is an example of a transitive dependency and a violation of Third Normal Form.



Third Normal Form Transitive Dependency

- The correctly normalized model is shown here:
create a second entity
STORE, with a relationship
to CD.



Third Normal Form Example

- Consider a system that tracks information about cities - size, population, mayor, and so on.
- The first model shows an entity that includes state information.
- Although state is an attribute of city, state flower is really an attribute of state.

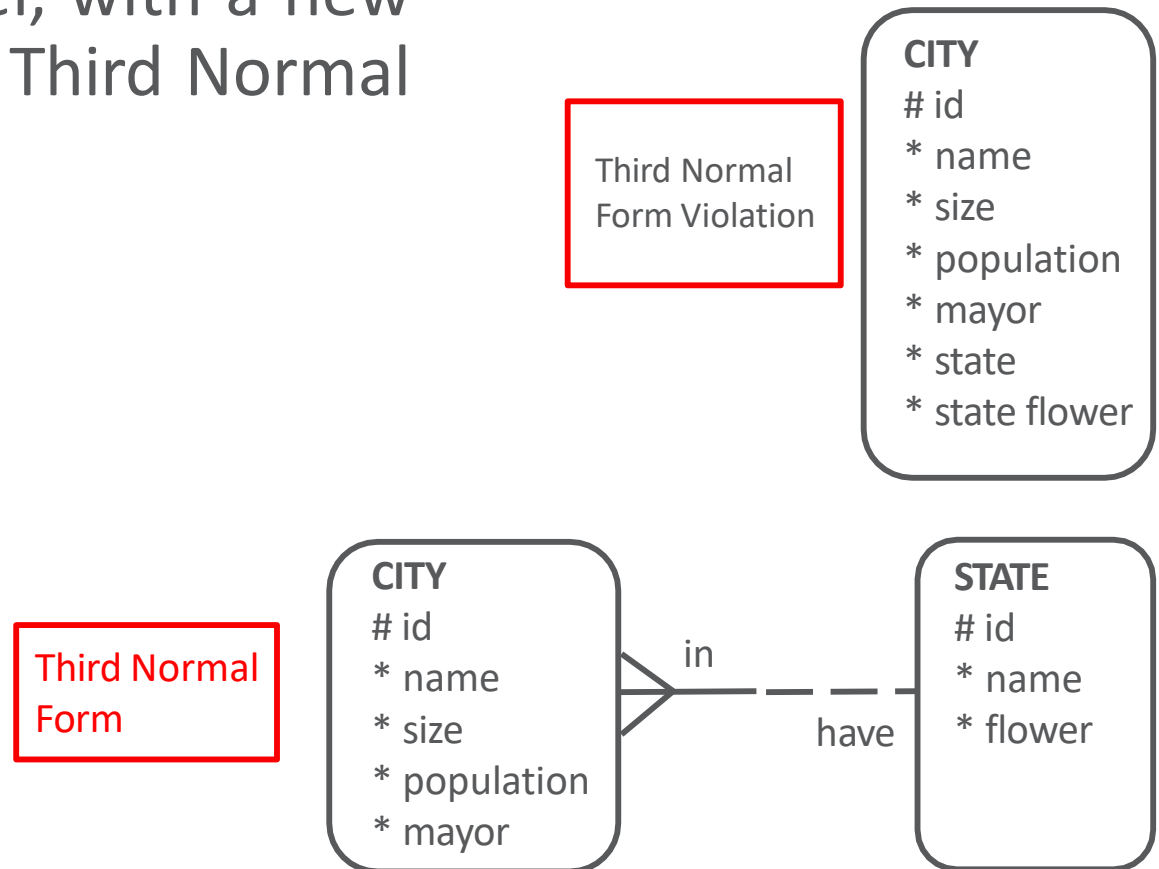
Third Normal
Form Violation

CITY

id
* name
* size
* population
* mayor
* state
* state flower

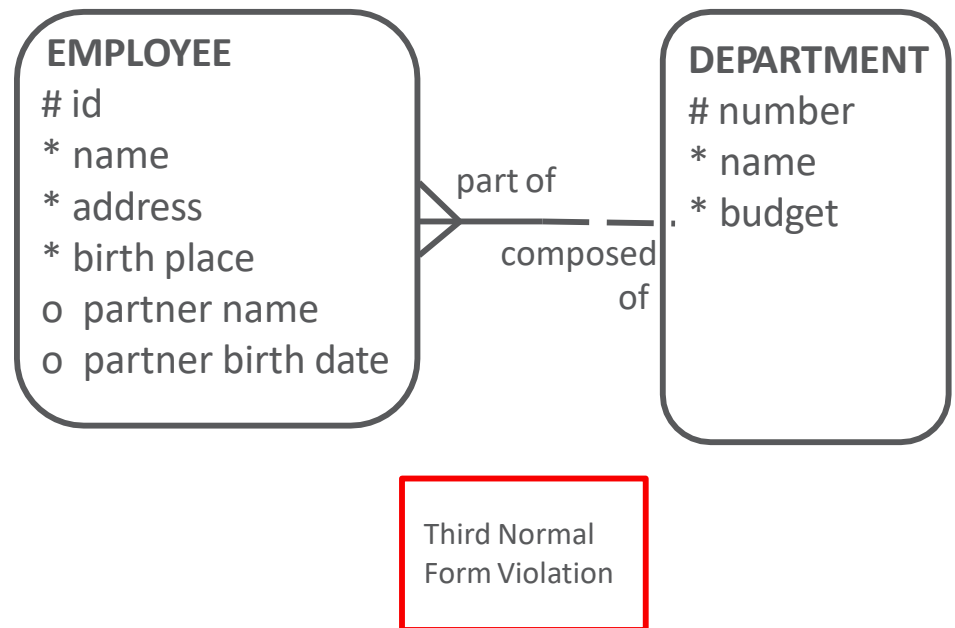
Third Normal Form Example

- The second model, with a new entity STATE, is in Third Normal Form.



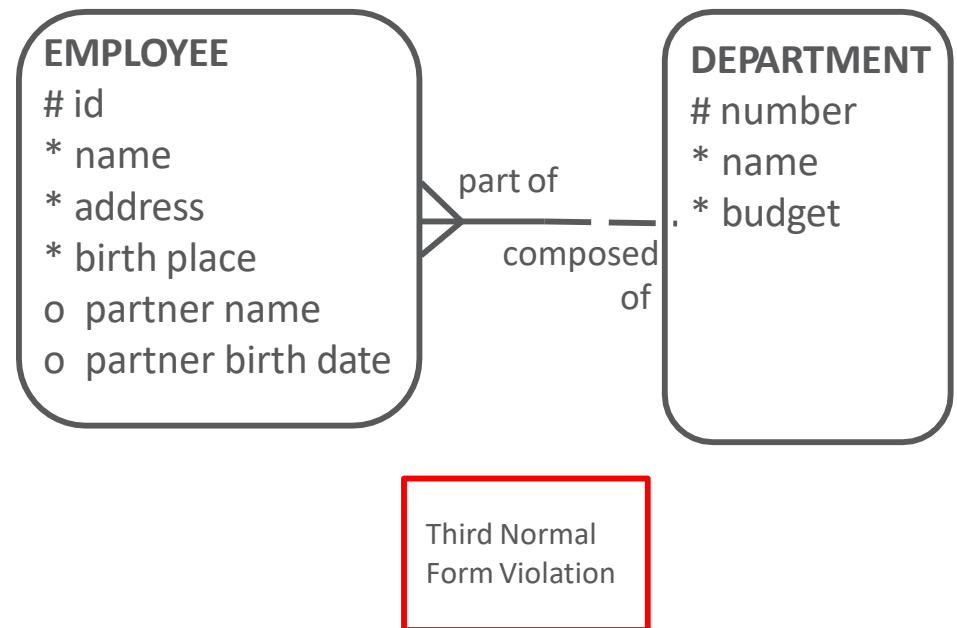
Third Normal Form Second Example

- In this example, assume the following business rule: each employee can have one partner.
- This model violates Third Normal Form because partner birth date is an attribute of partner, not of EMPLOYEE.



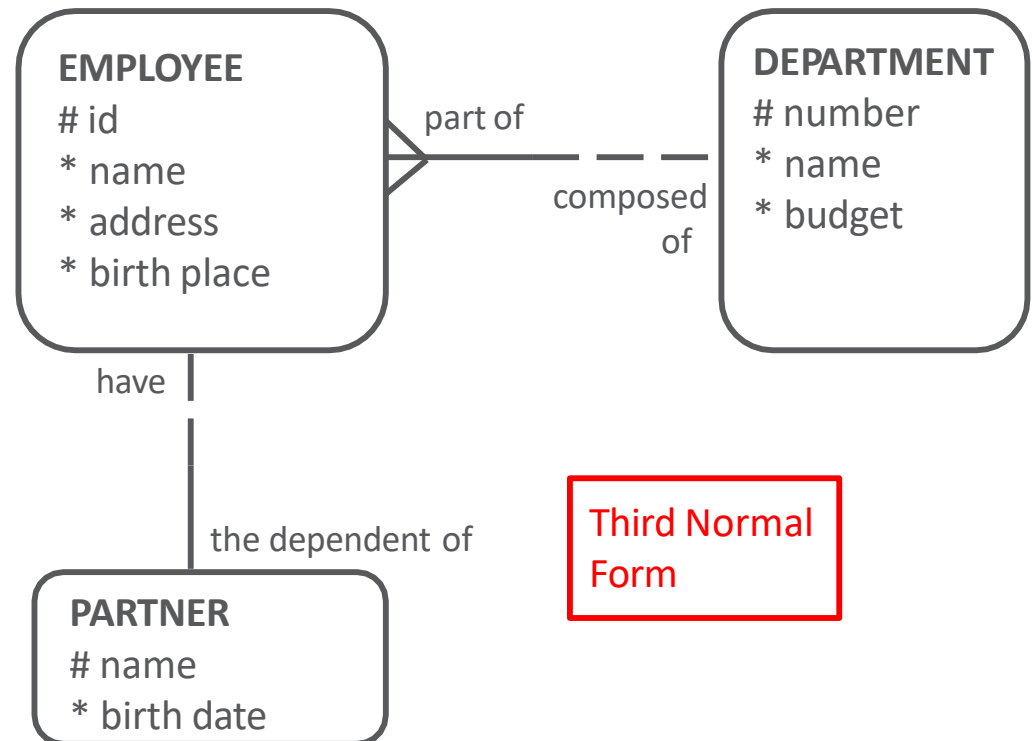
Third Normal Form Second Example

- Another way of stating Third Normal Form: non-UID attributes can't have attributes of their own.



3NF Second Example Solution

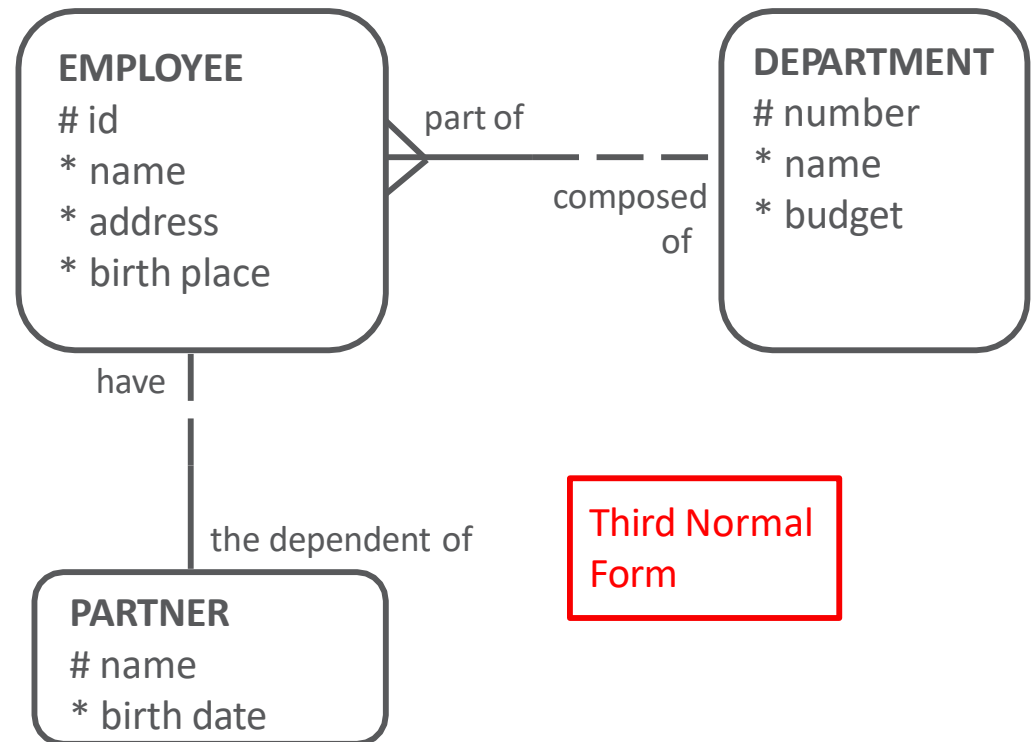
- This model supports Third Normal Form because partner birth date is an attribute of partner, not of EMPLOYEE.



Third Normal
Form

3NF Second Example Solution

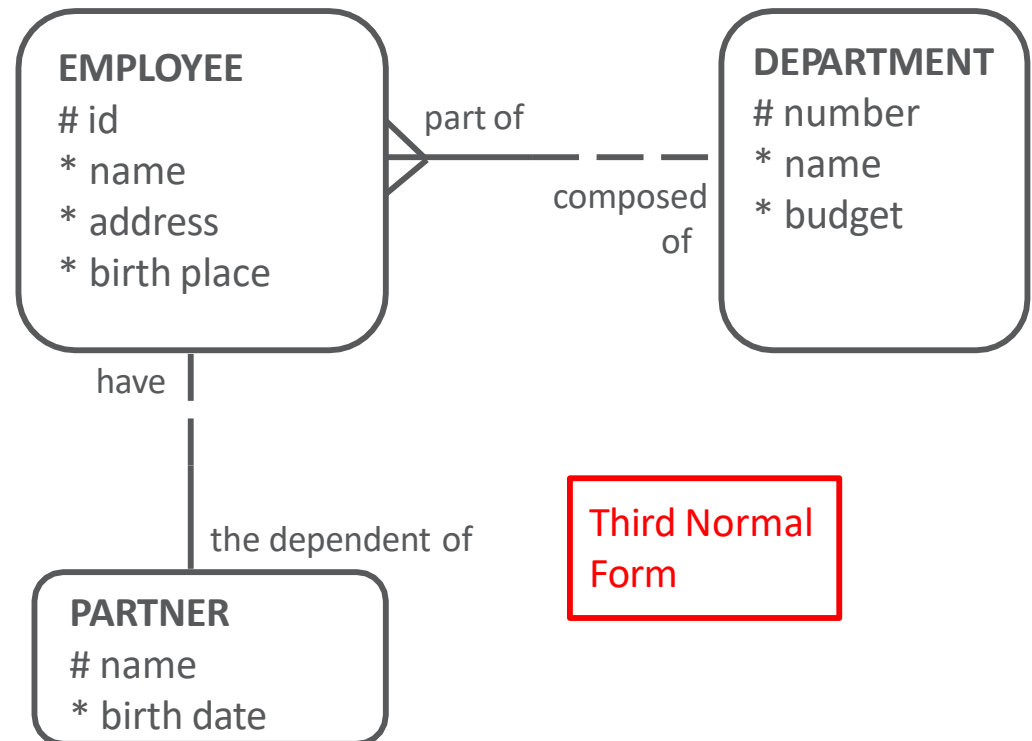
- The 1:1 relationship is optional on the EMPLOYEE end because some employees will not have partners.



Third Normal
Form

3NF Second Example Solution

- It is mandatory on the PARTNER end because information about a partner is tracked only if that person is a partner of one and only one EMPLOYEE.



Third Normal
Form

Terminology

Key terms used in this lesson included:

- Third Normal Form (3NF)
- Transitive dependency