IS475/675 Agenda: Week 6

- Finish discussing modeling data that will be stored over time (longitudinal, time-dependent data).
- Identify characteristics of a good design.
- Define normalization and learn the vocabulary of normalization.
- “Fix” a database design together based on the forms of normalization.
- Do an exercise and “fix” another database design.

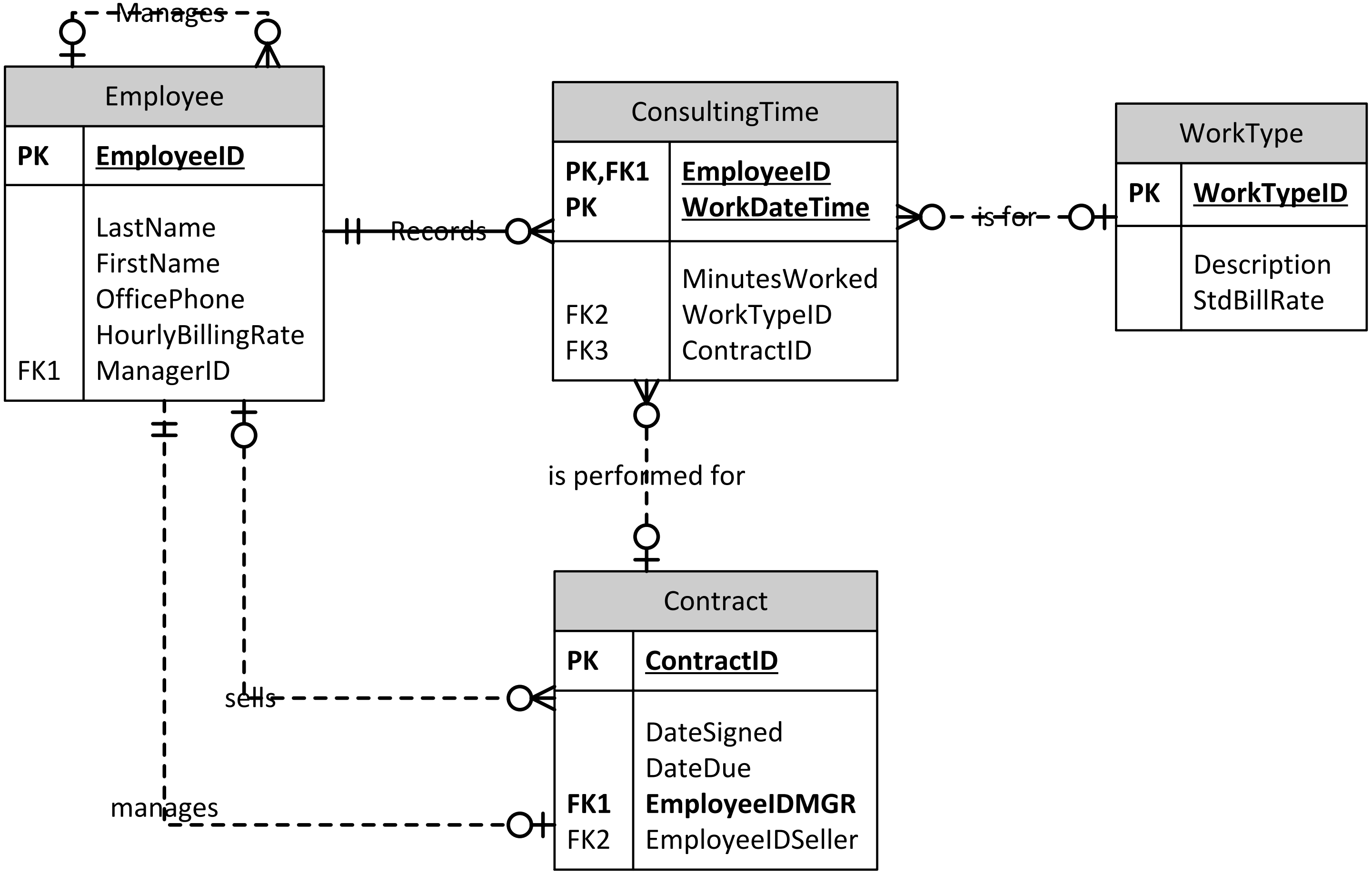
Review: what does it mean to store data over time?

- Data that must be maintained beyond a specific point in time.
- Examples:
  - Historical data that must be maintained to create decision-making information.
  - Current data that is updated on an ongoing basis, like a quantity on hand for inventory.
  - Current data that changes, like a person's pay rate, or a subscription type, or an employer.

Problem from HW#2

This was an ERD skeleton from HW#2. Design a database to help a consulting organization keep track of the amount of time an employee spends working on a contract. While working each day, each employee completes an online timesheet that looks like this:

Name	Tristan Elliott	**Date**	2/14/2025
Employee ID	3411		
Contract ID	**Type of Work Description**	**Time Start**	**Minutes Worked**
444	Python Programming	8AM	180
444	Tableau Report Generation	11AM	240
777	Tableau Report Generation	4PM	120


Manages
Employee
PK
EmployeeID
LastName
FirstName
OfficePhone
HourlyBillingRate
FK1
ManagerID
Records
ConsultingTime
PK,FK1
PK
EmployeeID
WorkDateTime
MinutesWorked
FK2
WorkTypeID
FK3
ContractID
is for
WorkType
PK
WorkTypeID
Description
StdBillRate
is performed for
sells
manages
Contract
PK
ContractID
DateSigned
DateDue
FK1
EmployeeIDMGR
FK2
EmployeeIDSeller

How long do we store data?

Name	Tristan Elliott	**Date**	2/14/2025
Employee ID	3411		
Contract ID	**Type of Work Description**	**Time Start**	**Minutes Worked**
444	Python Programming	8AM	180
444	Tableau Report Generation	11AM	240
777	Tableau Report Generation	4PM	120

What data do we store over time?

Storing data over time

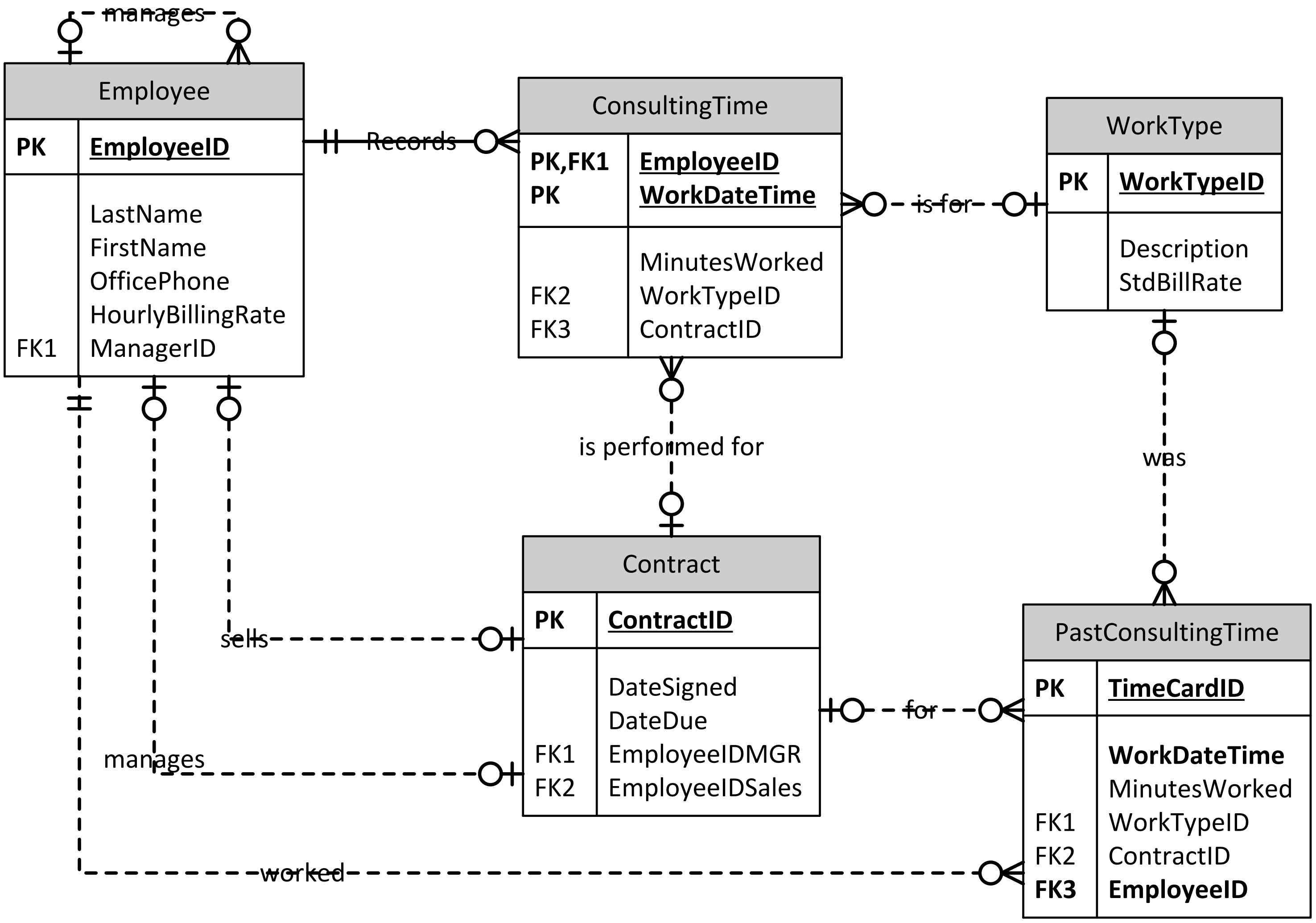
Consulting time data is associated with a given date and time (WorkDateTime).

How long should the data in that table be stored?

Where should the data be stored long-term, i.e. after the employee is paid or after the contract is completed?

Should the data be stored at the same level of detail as the original transaction (detailed level of "granularity") or should it be summarized?

Detailed level of granularity



Employee

PK	**EmployeeID**
	LastName FirstName OfficePhone HourlyBillingRate
FK1	ManagerID

ConsultingTime

PK,FK1	**EmployeeID**
PK	**WorkDateTime**
	MinutesWorked
FK2	WorkTypeID
FK3	ContractID

WorkType

PK	**WorkTypeID**
	Description StdBillRate

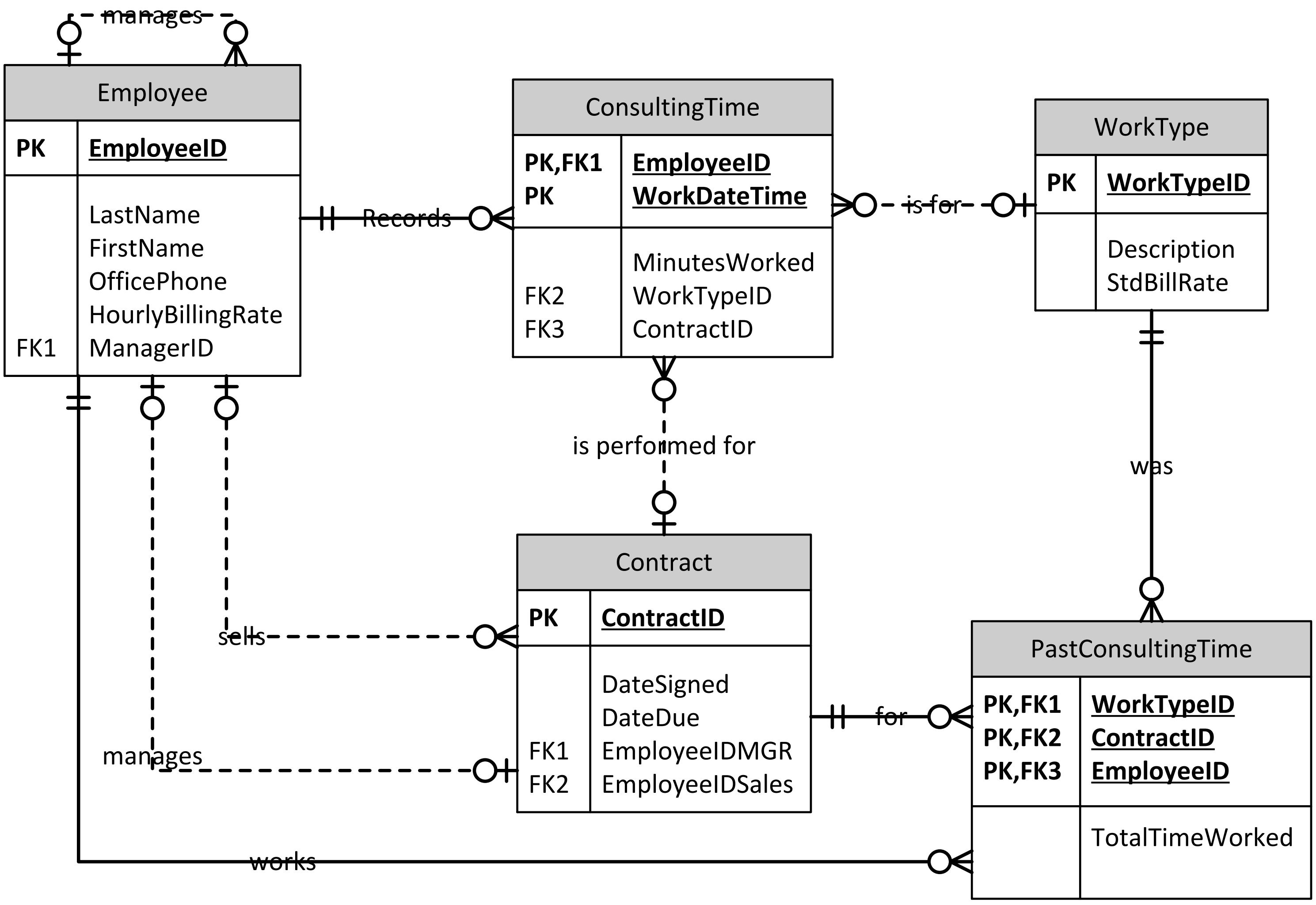
Contract

PK	**ContractID**
	DateSigned DateDue
FK1	EmployeeIDMGR
FK2	EmployeeIDSales

PastConsultingTime

PK	**TimeCardID**
	WorkDateTime MinutesWorked
FK1	WorkTypeID
FK2	ContractID
FK3	**EmployeeID**

Summarized level of granularity by Employee, WorkType and Contract



Employee

PK **EmployeeID**

LastName
FirstName
OfficePhone
HourlyBillingRate
FK1 ManagerID

ConsultingTime

PK,FK1 **EmployeeID**
PK **WorkDateTime**

MinutesWorked
FK2 WorkTypeID
FK3 ContractID

WorkType

PK **WorkTypeID**

Description
StdBillRate

Contract

PK **ContractID**

DateSigned
DateDue
FK1 EmployeeIDMGR
FK2 EmployeeIDSales

PastConsultingTime

PK,FK1 **WorkTypeID**
PK,FK2 **ContractID**
PK,FK3 **EmployeeID**

TotalTimeWorked

What are the decisions for a database designer?

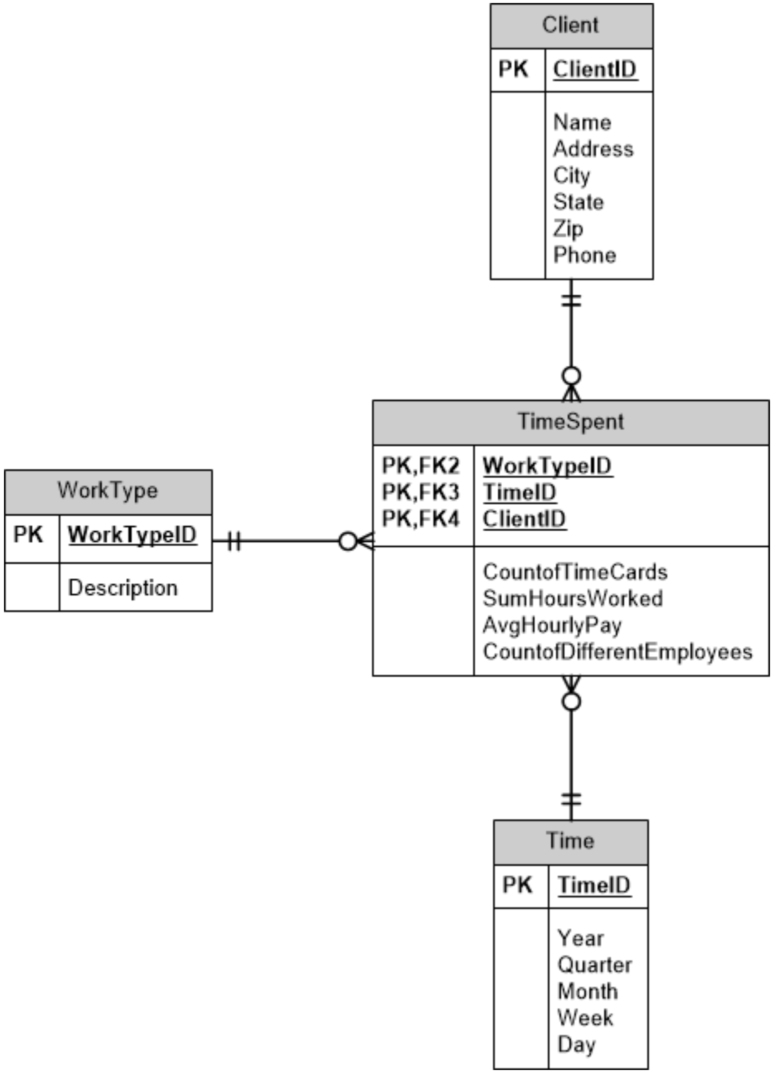
Should “past” and “current” data be stored together in a single table?

Should “past” and “current” data be stored together in a single database?

What level of granularity is required for the “past” data?

When should the data be moved from the “current” tables to the “past” tables?

Sample data mart for consulting time



Client	
PK	**ClientID**
	Name Address City State Zip Phone

TimeSpent	
PK,FK2 PK,FK3 PK,FK4	**WorkTypeID** **TimeID** **ClientID**
	CountofTimeCards SumHoursWorked AvgHourlyPay CountofDifferentEmployees

WorkType	
PK	**WorkTypeID**
	Description

Time	
PK	**TimeID**
	Year Quarter Month Week Day

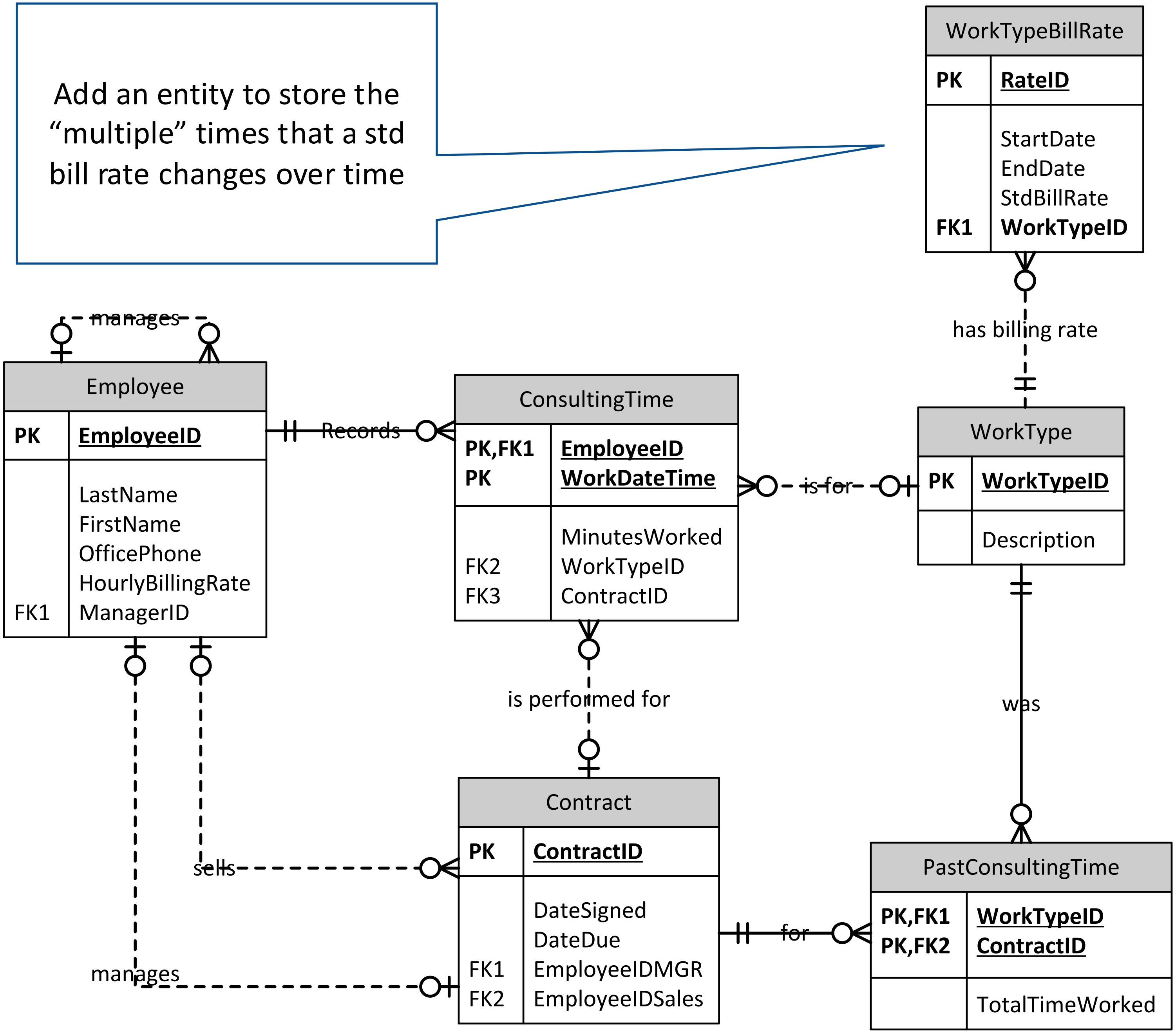
What happens when data changes over time and you want to store the changing data?

- What if you want to keep track of the stdbillrate for a worktype over a period of time?

Time Period	Work Type	Std Bill Rate
1/1/2020 – 8/10/2021	255	$125
8/11/2021 – 12/31/2021	255	$135
1/1/2022 – 12/31/2022	255	$125
1/1/2023 – 01/15/2024	255	$130
1/16/2024– now	255	$175

Slowly changing dimension

- Master data, such as customer or employee data is relatively static.
- Sometimes the data values will change over time. Examples are StdBillRate in the Work Type entity, or HourlyBillingRate in the Employee entity.
- Want to keep a history of the rates rather than overwriting the rates as they change.
- An attribute which changes over time in a master data entity is referred to as a "slowly changing dimension" of that entity.



Add an entity to store the "multiple" times that a std bill rate changes over time

WorkTypeBillRate	
PK	**<u>RateID</u>**
	StartDate EndDate StdBillRate
FK1	**WorkTypeID**

Employee	
PK	**<u>EmployeeID</u>**
	LastName FirstName OfficePhone HourlyBillingRate
FK1	ManagerID

ConsultingTime	
PK,FK1	**<u>EmployeeID</u>**
PK	**<u>WorkDateTime</u>**
	MinutesWorked
FK2	WorkTypeID
FK3	ContractID

WorkType	
PK	**<u>WorkTypeID</u>**
	Description

Contract	
PK	**<u>ContractID</u>**
	DateSigned DateDue
FK1	EmployeeIDMGR
FK2	EmployeeIDSales

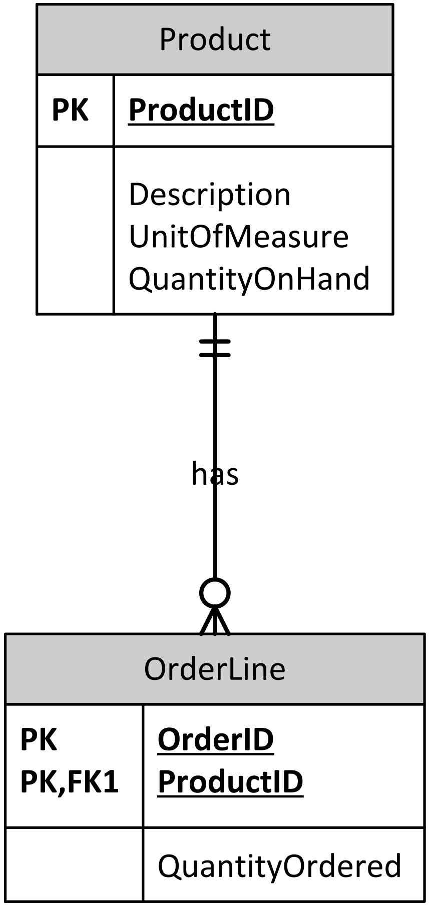
PastConsultingTime	
PK,FK1	**<u>WorkTypeID</u>**
PK,FK2	**<u>ContractID</u>**
	TotalTimeWorked

Modeling Time-Dependent Data

- Status vs. event data
  - **Status** data gives a view of data at a particular point-in-time. Example: Product data that includes a current quantity on hand.

  - **Event** data is a transaction that occurs that changes the status data. For example, if an order is placed for a given product, the quantity on hand will likely decrease.

  - The database designer must decide how to store the status data, and how to store the event data.



Product

PK ProductID

Description
UnitOfMeasure
QuantityOnHand

has

OrderLine

PK OrderID
PK,FK1 ProductID

QuantityOrdered

Product Data: Status Data

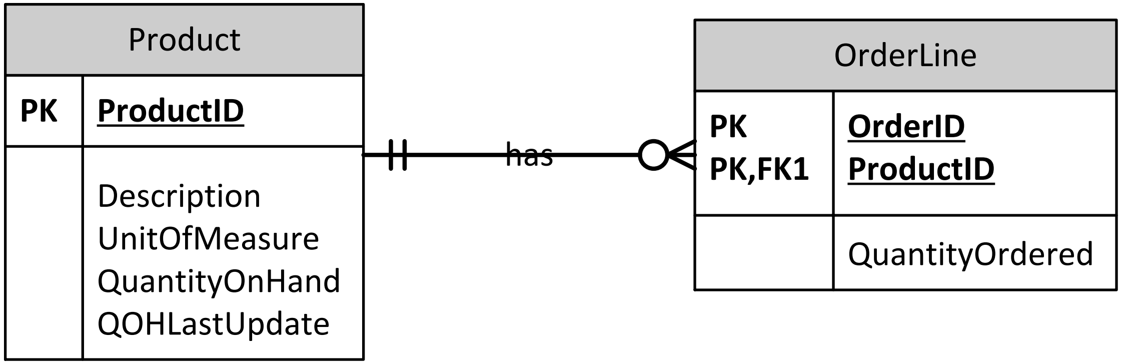
ProductID	Description	Unit of Measure	Quantity on Hand
1234-5	36” Computer Desk	Each	20
1234-8	48” Computer Desk	Each	5

Order Line Data: Event Data

OrderID	ProductID	Quantity Ordered
100	1234-5	2
100	1234-8	3

When did the event occur? Does the quantity on hand reflect the event? What are the problems involved with storing a field like “quantity on hand” with the product status data?

Event and status data must include a time stamp (a date/time field) to reconcile the status with the event.



Product

PK ProductID

Description
UnitOfMeasure
QuantityOnHand
QOHLastUpdate

has

OrderLine

PK OrderID
PK,FK1 ProductID

QuantityOrdered

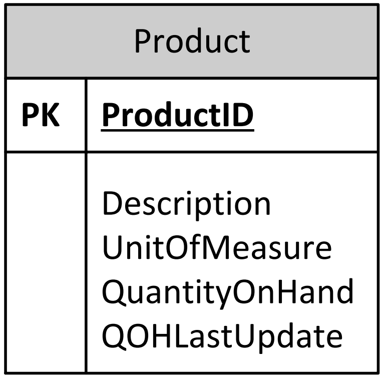
ProductID	Description	Unit of Measure	Quantity on Hand	QOH LastUpdate
1234-5	36” Computer Desk	Each	20	2/20/2025 09:00:02
1234-8	48” Computer Desk	Each	5	2/23/2025 21:15:12

OrderID	ProductID	Quantity Ordered
100	1234-5	2
100	1234-8	3

Current data that changes over time.

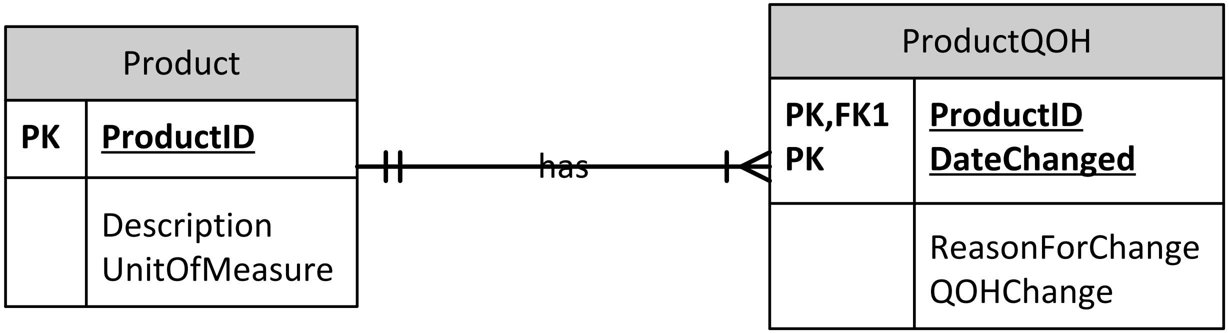
Transient vs. Periodic data: This difference is related to the processing of data.

- **Transient** data is data that will be overwritten when a change occurs to the data. For example, if we overwrite the quantity on hand, then the quantity on hand is "transient." In SQL, this is an "update" operation.
- **Periodic** data are never changed – we maintain a history of all the changes that occur. The changes would be a multi-valued attribute so they are stored in a separate entity. For example, if we keep track of each change to the quantity on hand, then the quantity on hand is actually periodic data. In SQL, this is an "insert" operation.



Product	
PK	**ProductID**
	Description UnitOfMeasure QuantityOnHand QOHLastUpdate

Transient data with time stamp



Product	
PK	**ProductID**
	Description UnitOfMeasure

ProductQOH	
PK,FK1 **PK**	**ProductID** **DateChanged**
	ReasonForChange QOHChange

Periodic data with time stamp and field used to calculate the actual QOH (quantity on hand)

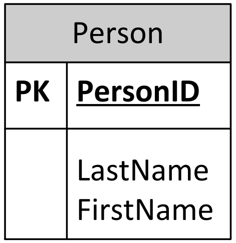
Let’s do another design!

Design a database to keep track of the locations where a person has lived. For each person, we want to keep track of his/her personID (unique identifier), last name and first name.

For each person’ location, we want to keep track of the start date and end date that the person lived in a place, the address of the location, and the type of location it is (i.e. house, apartment, yurt, condo).

If a person lives in a house, we also want to keep track of the square footage of the house. If a person lives in a multi-family dwelling, we want to keep track of the number of families that could live in that dwelling.

I’ll start you out!



Person	
PK	**<u>PersonID</u>**
	LastName FirstName

What makes a good design?

There is no redundancy in the rows of a single table/entity.

There is no redundancy in the rows of different tables/entities in the database.

Each of the attributes inside a given table/entity have one data value.

Each of the attributes inside a given table/entity belong together.

Design will be implemented in a relational database.

What are the characteristics of a relational database?

- All data is stored in two-dimensional tables.
- Tables are not related to each other hierarchically.
- Tables cannot inherit data from other tables.
- In most relational DBMS's there is no generalization or specialization types of relationships available.
- Tables can only be related to each other one-table-to-one-table on the basis of a foreign key.

More characteristics of a relational database

- Each column in a table must be of one primitive data type (i.e. character, integer, floating point, fixed decimal point, date, datetime, variable character)
- The quantity, data types and names of columns are established when a table is created. It is relatively difficult to add more columns.
- The quantity of rows is variable. It is very easy to add more rows.
- The intersection of a column and row (cell) can store only one value.
- The primary key (one or more columns) must contain a data value that is unique among all rows.

What is normalization?

- Normalization is a formal, process-oriented approach to data modeling.
- Normalization helps the designer understand the functional relationships among data attributes.
- We have been focusing on relationships between entities, but normalization focuses on relationships between attributes in an entity.
- A set of related attributes is called a “relation.”

Vocabulary for normalization

- A “functional dependency" is a **relationship** between attributes in which one attribute or group of attributes determines the value of another.

- A “determinant” is an attribute or group of attributes that, once known, can determine the value of another attribute.

Examples of functional dependencies and determinants

- A social security number determines your name and address.
  SSN → name, address.
- A vehicle id number determines the make and model of a car.
  VIN → make, model.
- Name and address are "functionally dependent" on SSN.
- SSN "determines" name and address.

Functional dependencies

- Functional dependency diagram format:
  - CourseID → CourseName, CourseDescription, CourseCredits
  - ZipCode → City, State
  - PatientID, TreatmentID, TreatmentDateTime → TestResults
  - ArtItemCode, StartDateTime → Location
- Natural determinants help highlight functional dependencies.
- Surrogate determinants don’t help the designer understand the functional dependencies.
- Must make decisions about the data.
  - Are course credits above really dependent on the courseID?
  - What else could course credits be dependent on?

Normalization process

- Normalization is accomplished in stages. A "normal form" is a state (level of completeness) of a data model.

- Unnormalized data: A data model that has not been normalized. It is not a stable model and will not be adaptable to change.

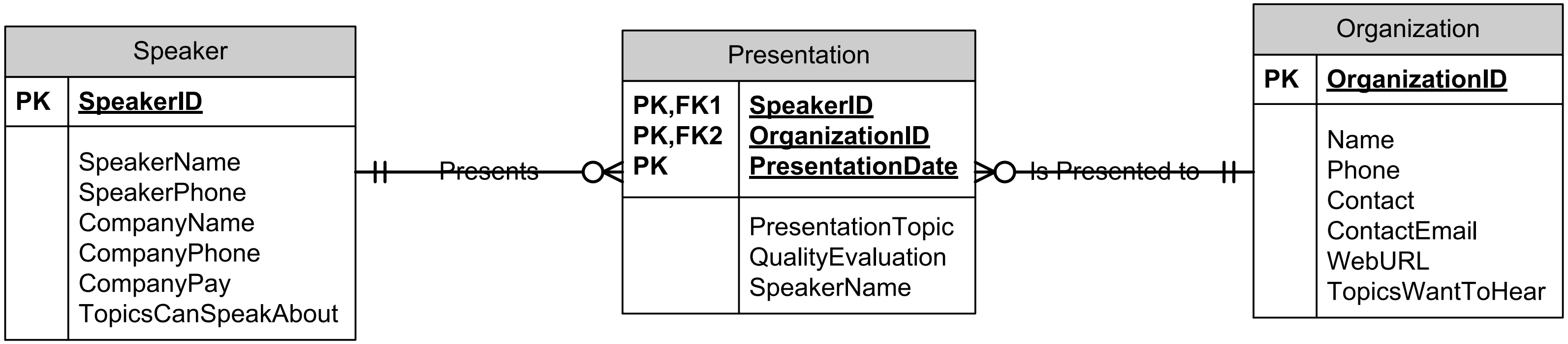
- Unnormalized data is essentially one or a couple of entities.

- Unnormalized data usually has significant data redundancy.

Steps in Normalization

- There are three basic “problems” that a designer identifies with a data model:
    1. Multi-valued attributes (also called “repeating groups” of attributes).
    2. Partial functional dependencies.
    3. Transitive dependencies.
- For each problem, the designer takes the “problem” data and puts it into a separate entity.
- The designer creates an appropriate primary key and relates the new entity to the existing entity with a foreign key.

Exercise example – speaker database



Speaker	
PK	**SpeakerID**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay TopicsCanSpeakAbout

Presentation	
PK,FK1 **PK,FK2** **PK**	**SpeakerID** **OrganizationID** **PresentationDate**
	PresentationTopic QualityEvaluation SpeakerName

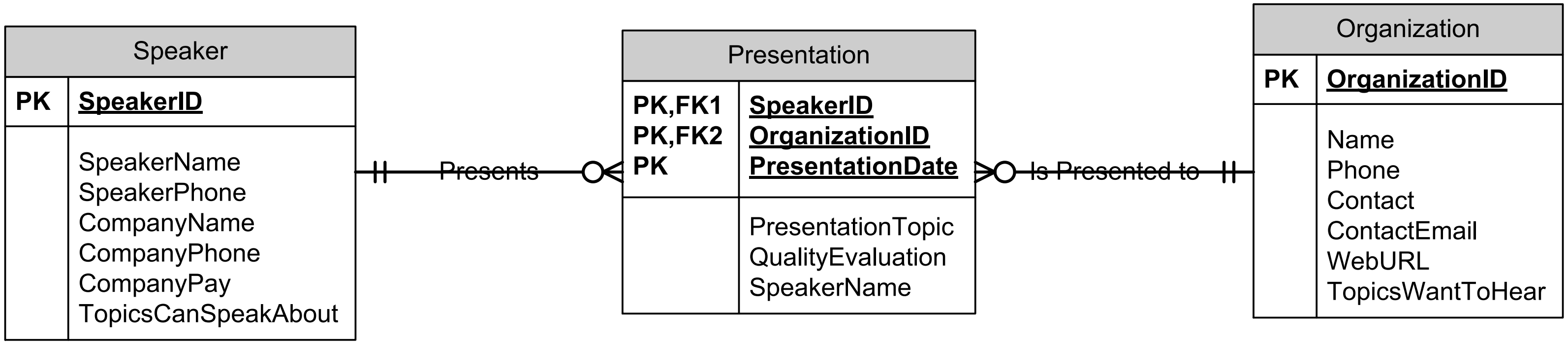
Organization	
PK	**OrganizationID**
	Name Phone Contact ContactEmail WebURL TopicsWantToHear

The application is a database to keep track of speakers giving presentations for student organizations on campus. Student organizations, such as Beta Alpha Psi, Business Student Council, Cyber Security Center, Economics Club, etc. have meetings where they bring in speakers from the business community to give presentations about various topics. The student organization leaders decided it would be helpful to have a centralized database of all presentations given to all student organizations so that they wouldn’t duplicate speakers or topics too often. They also thought it would be great to have a way to find the names of potential speakers for future meetings.

Speaker ID	Speaker Name	Speaker Phone	Company Name	Company Phone	Company Pay	Topics to present
12	Beth Adams	775-233-4444	SNC	775-652-8991	No	• ERP Systems • Social Media Marketing • FinTech: Blockchain in financial transactions

Organization ID	Name	Contact	Contact Email	WebURL	Topics to hear
3	Business Student Council	James Doe	doe@unr.edu	www.dgs.org	• Career Skills • Social Media Marketing • FinTech • Big Data & Decision making

Speaker ID	Organization ID	Presentation Date	Presentation Topic	Quality Evaluation	Speaker Name
12	3	9/27/2024	Social Media Marketing	Clear discussion but not very dynamic or energetic. Good examples, but boring.	Beth Adams



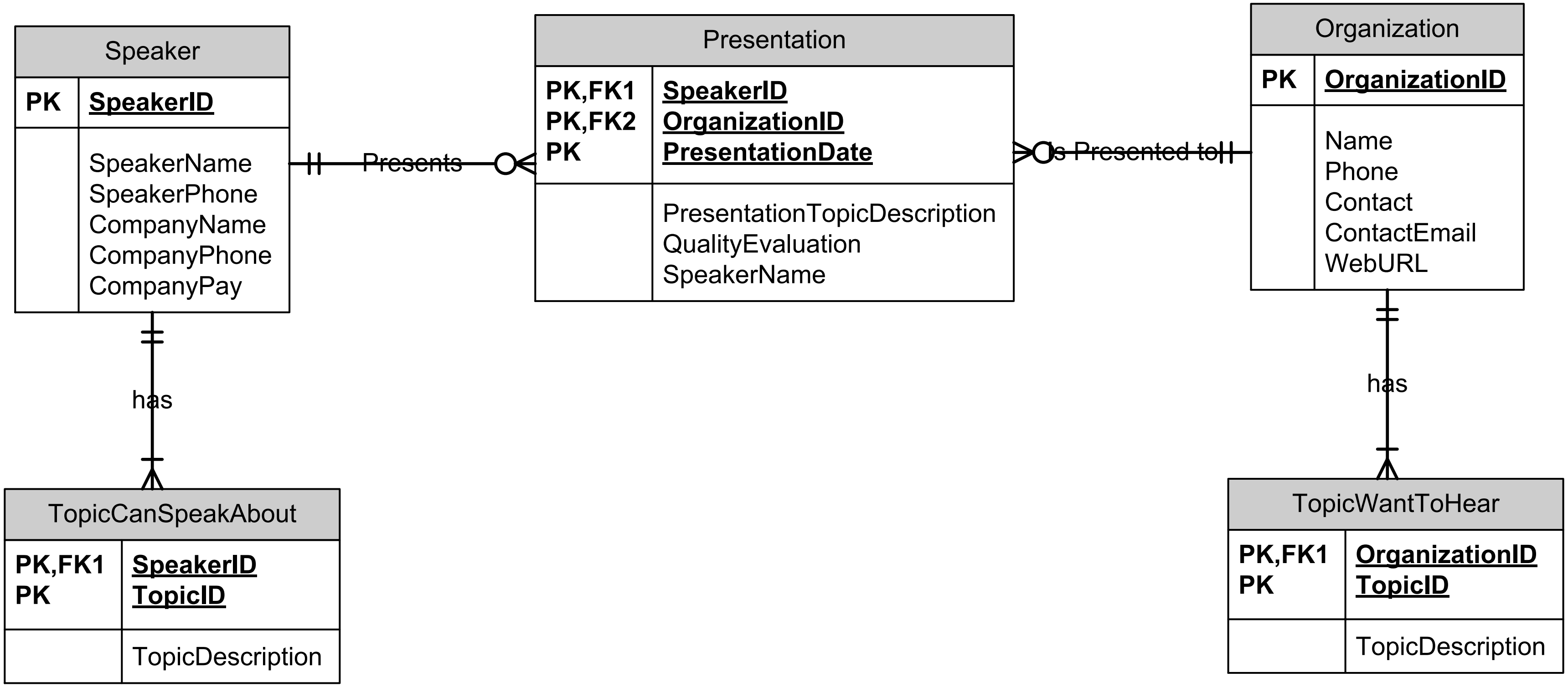
Speaker	
PK	**<u>SpeakerID</u>**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay TopicsCanSpeakAbout

Presentation	
PK,FK1 **PK,FK2** **PK**	**<u>SpeakerID</u>** **<u>OrganizationID</u>** **<u>PresentationDate</u>**
	PresentationTopic QualityEvaluation SpeakerName

Organization	
PK	**<u>OrganizationID</u>**
	Name Phone Contact ContactEmail WebURL TopicsWantToHear

First Normal Form

- First normal form: Remove multi-valued attributes.
  - A multi-valued attribute is an attribute or group of attributes that can have more than one value for an instance of an entity. If it is a group of attributes, it is called a “repeating group.”
- To see whether a data model is in first normal form:
  - Identify repeating groups/multi-valued attributes and place them as separate entities in the model.
- How to fix the problem:
  - Identify a primary key for the new entity. The primary key will most likely be concatenated.
  - Create the relationships between entities.
  - Divide m:n relationships with appropriate intersection entities.



Speaker	
PK	**SpeakerID**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay

Presentation	
PK,FK1 PK,FK2 PK	**SpeakerID** **OrganizationID** **PresentationDate**
	PresentationTopicDescription QualityEvaluation SpeakerName

Organization	
PK	**OrganizationID**
	Name Phone Contact ContactEmail WebURL

TopicCanSpeakAbout	
PK,FK1 PK	**SpeakerID** **TopicID**
	TopicDescription

TopicWantToHear	
PK,FK1 PK	**OrganizationID** **TopicID**
	TopicDescription

SpeakerID -> SpeakerName, SpeakerPhone, CompanyName, CompanyPhone, CompanyPay

SpeakerID, OrganizationID, PresentationDate -> PresentationTopic, QualityEvaluation, SpeakerName

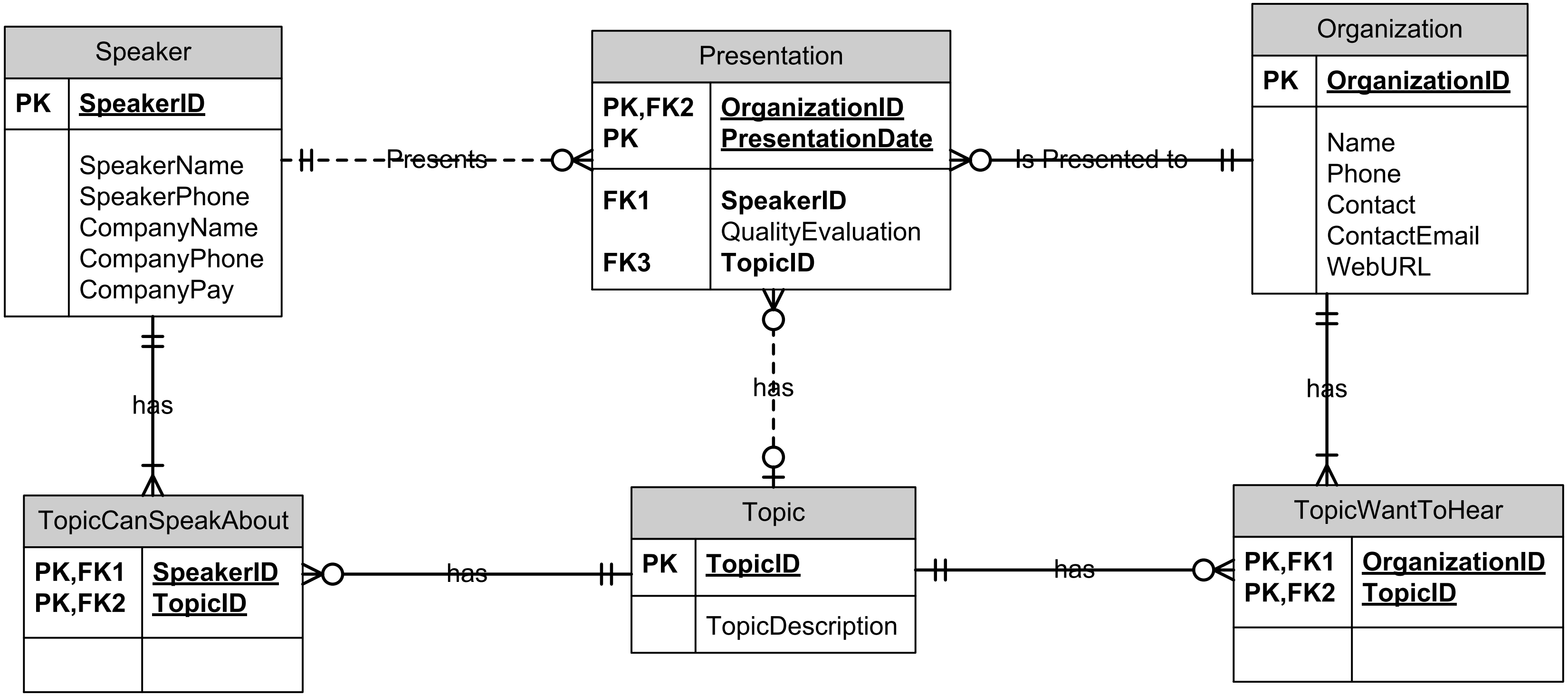
OrganizationID -> Name, Phone, Contact, ContactEmail, WebURL

SpeakerID, TopicID -> TopicDescription

OrganizationID, TopicID -> TopicDescription

Second Normal Form

- Second normal form: Remove partial functional dependencies.
- A partial functional dependency is a situation in which one or more non-key attributes are functionally dependent on part, but not all, of the primary key.
  - Partial functional dependencies occur only with concatenated keys.
- Examples of partial functional dependencies:
  - PatientID, TreatmentDateTime → PatName, TstResults, TrtID, LocID
  - CourseID, StudentID → CourseTitle, Grade
- Which entities on the current ERD have a concatenated primary key?



Speaker

PK	**SpeakerID**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay

Presentation

PK,FK2 PK	**OrganizationID** **PresentationDate**
FK1 FK3	**SpeakerID** QualityEvaluation **TopicID**

Organization

PK	**OrganizationID**
	Name Phone Contact ContactEmail WebURL

TopicCanSpeakAbout

PK,FK1 PK,FK2	**SpeakerID** **TopicID**

Topic

PK	**TopicID**
	TopicDescription

TopicWantToHear

PK,FK1 PK,FK2	**OrganizationID** **TopicID**

SpeakerID -> SpeakerName, SpeakerPhone, CompanyName, CompanyPhone, CompanyPay

OrganizationID, PresentationDate -> SpeakerID, QualityEvaluation, TopicID

OrganizationID -> Name, Phone, Contact, ContactEmail, WebURL

TopicID -> TopicDescription

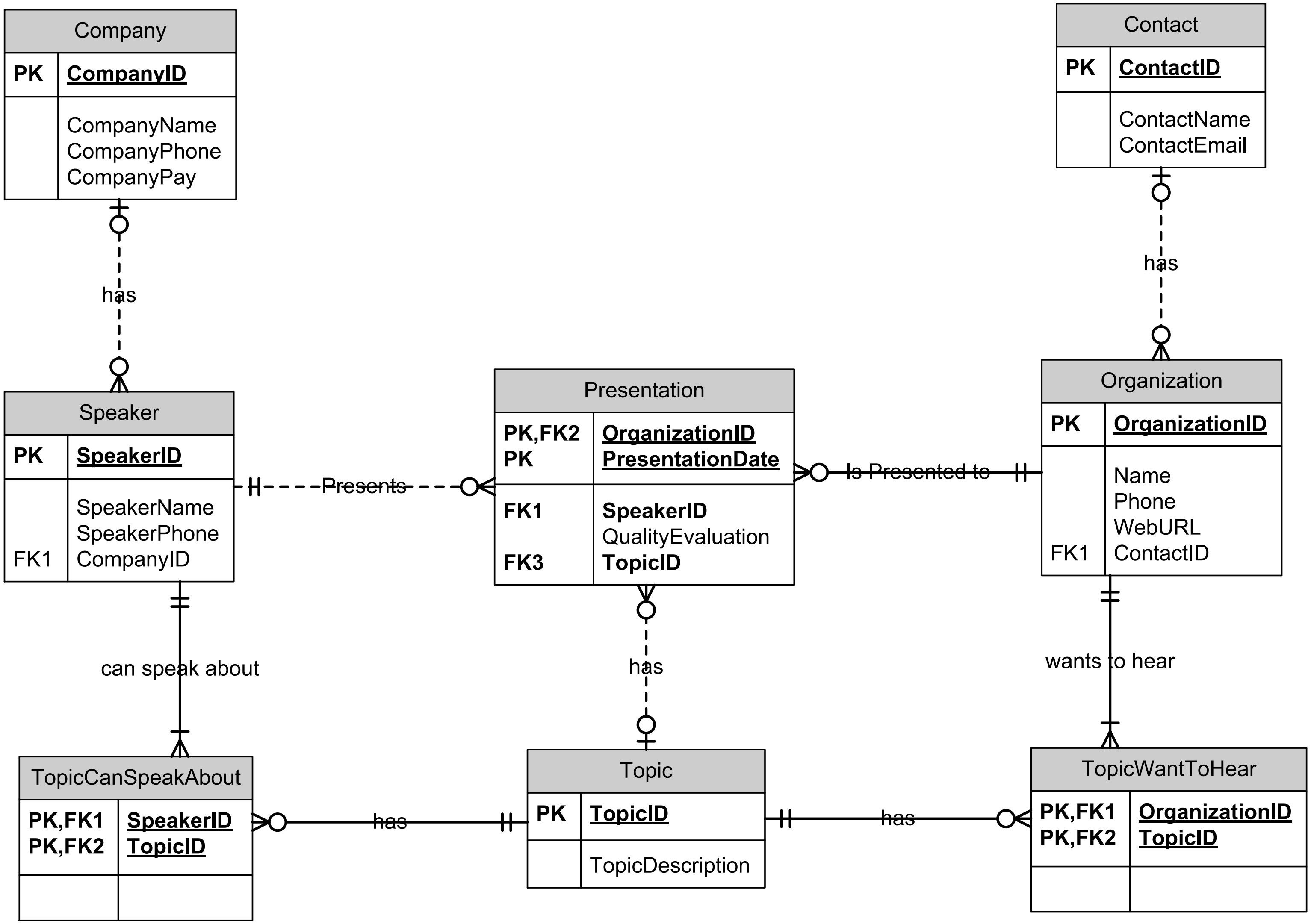
SpeakerID, TopicID ->

OrganizationID, TopicID ->

Third normal form

- Third normal form: Remove transitive dependencies.
  - A transitive dependency occurs when a non-key attribute is functionally dependent on one or more non-key attributes.
- Third normal form examines entities with single primary keys and removes the “floating” or transitive dependencies.
- It may be possible to have attributes that are determined by other attributes, rather than by the primary key. They must be removed into entities with appropriate primary keys.
- Example of transitive dependency:

??->CompanyName, CompanyPhone, CompanyPay



Third Normal Form

Functional Dependency Diagrams

CompanyID -> CompanyName, CompanyPhone, CompanyPay

SpeakerID -> SpeakerName, SpeakerPhone, CompanyID

ContactID -> ContactName, ContactEmail

OrganizationID, PresentationDate -> SpeakerID, QualityEvaluation, TopicID

OrganizationID -> Name, Phone, ContactID, WebURL

TopicID -> TopicDescription

SpeakerID, TopicID ->

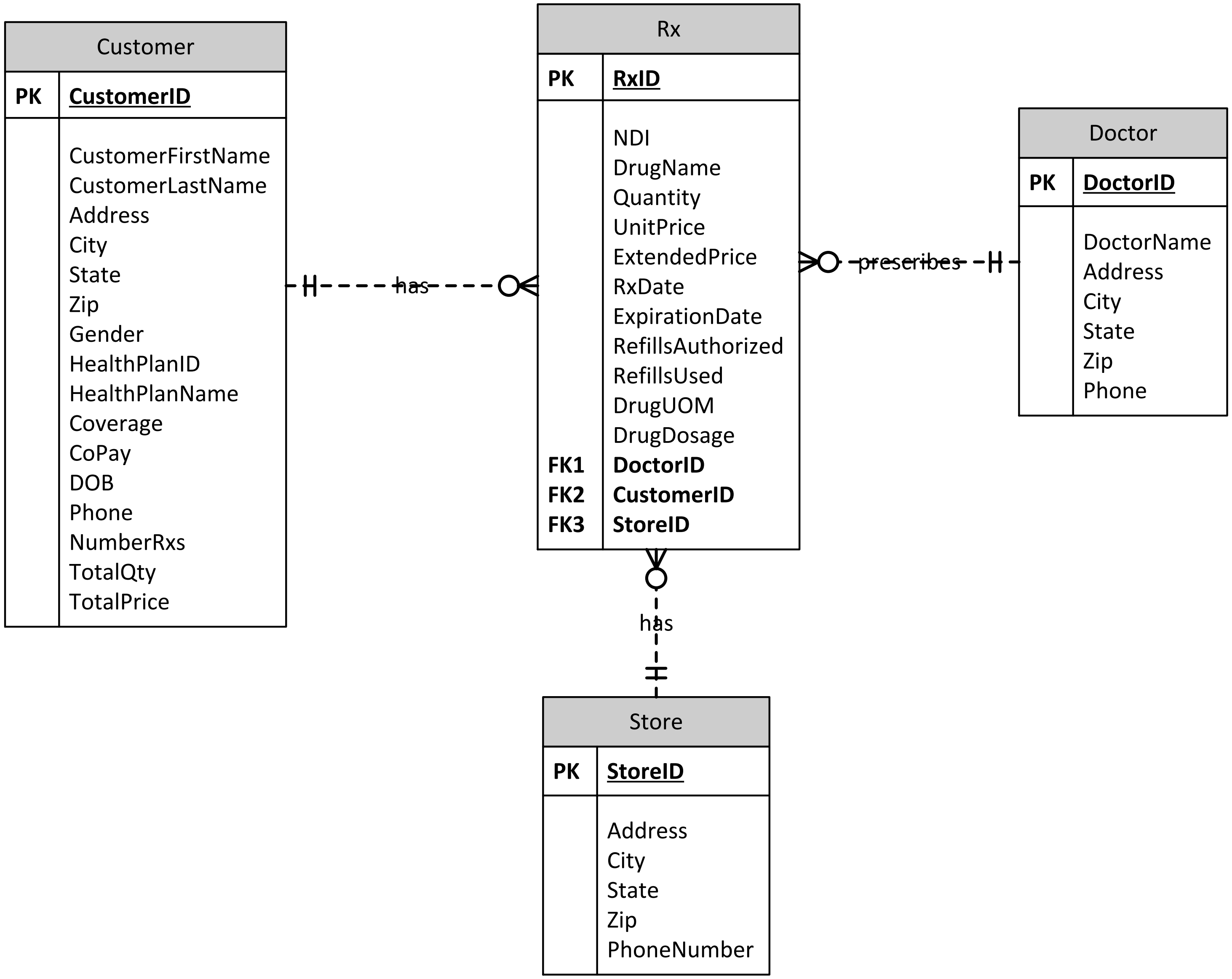
OrganizationID, TopicID ->

Summary of normalization process

- Examine and evaluate the logical data model:
  - Find the **repeating groups and/or multi-valued attributes** and put the model into first normal form. Identify primary key fields for any new entities. Create concatenated keys as necessary. Relate entities with foreign keys.
  - Find the **functional dependencies**. Identify the **partial functional dependencies** and put the model into second normal form. Identify primary key fields for any new entities. Create concatenated keys as necessary. Relate entities with foreign keys.
  - Find the **transitive dependencies** and put the model into third normal form. Identify primary key fields for any new entities. Relate entities with foreign keys.

The goals of this exercise are to evaluate and fix an ERD based on the principles of normalization.

- First, examine the ERD and identify any multi-valued attributes. Ask yourself if any of the data in the existing entities will have more than one value for a given entity instance. If you see a group of multi-valued attributes (repeating group), put the data that forms the repeating group into its own entity with an applicable primary key. The primary key of the new entity will most likely be concatenated.
- Second, examine the ERD and look only at the entities with concatenated primary keys. Make sure that each attribute within an entity with a concatenated primary key is dependent on the whole primary key, rather than just part of the primary key. For any attribute that is not dependent on the whole primary key, remove that attribute from the entity and put it into an entity where it is dependent on the whole primary key. Make a new entity if necessary. The primary key of the new entity you create may or may not be concatenated.
- Third, examine the ERD and look at each entity. Check to make sure that each and every attribute within the entity is dependent on the primary key, and not some other attribute within the entity or the primary key of another entity. If such a “transitive dependency” occurs, then remove that attribute from the entity and put it in another entity where it is dependent on the primary key. Make a new entity if necessary.



Customer	
PK	**CustomerID**
	CustomerFirstName CustomerLastName Address City State Zip Gender HealthPlanID HealthPlanName Coverage CoPay DOB Phone NumberRxs TotalQty TotalPrice

Rx	
PK	**RxID**
	NDI DrugName Quantity UnitPrice ExtendedPrice RxDate ExpirationDate RefillsAuthorized RefillsUsed DrugUOM DrugDosage
FK1	**DoctorID**
FK2	**CustomerID**
FK3	**StoreID**

Doctor	
PK	**DoctorID**
	DoctorName Address City State Zip Phone

Store	
PK	**StoreID**
	Address City State Zip PhoneNumber

IS475/675 Agenda: Week 6

Finish discussing modeling data that will be stored over time (longitudinal, time-dependent data).

Identify characteristics of a good design.

Define normalization and learn the vocabulary of normalization.

“Fix” a database design together based on the forms of normalization.

Do an exercise and “fix” another database design.

What makes a good design?

There is no redundancy in the rows of a single table/entity.

There is no redundancy in the rows of different tables/entities in the database.

Each of the attributes inside a given table/entity have one data value.

Each of the attributes inside a given table/entity belong together.

What is normalization?

- Normalization is a formal, process-oriented approach to data modeling.
- Normalization helps the designer understand the functional relationships among data attributes.
- We have been focusing on relationships between entities, but normalization focuses on relationships between attributes in an entity.
- A set of related attributes is called a “relation.”

Vocabulary for normalization

- A “functional dependency" is a **relationship** between attributes in which one attribute or group of attributes determines the value of another.

- A “determinant” is an attribute or group of attributes that, once known, can determine the value of another attribute.

Examples of functional dependencies and determinants

- A social security number determines your name and address.
  SSN → name, address.
- A vehicle id number determines the make and model of a car.
  VIN → make, model.
- Name and address are “functionally dependent” on SSN.
- SSN “determines” name and address.

Functional dependencies

- Functional dependency diagram format:
    - CourseID → CourseName, CourseDescription, CourseCredits
    - ZipCode → City, State
    - PatientID, TreatmentID, TreatmentDateTime → TestResults
    - ArtItemCode, StartDateTime → Location
- Natural determinants help highlight functional dependencies.
- Surrogate determinants don't help the designer understand the functional dependencies.
- Must make decisions about the data.
    - Are course credits above really dependent on the courseID?
    - What else could course credits be dependent on?

Normalization process

- Normalization is accomplished in stages. A “normal form” is a state (level of completeness) of a data model.

- Unnormalized data: A data model that has not been normalized. It is not a stable model and will not be adaptable to change.

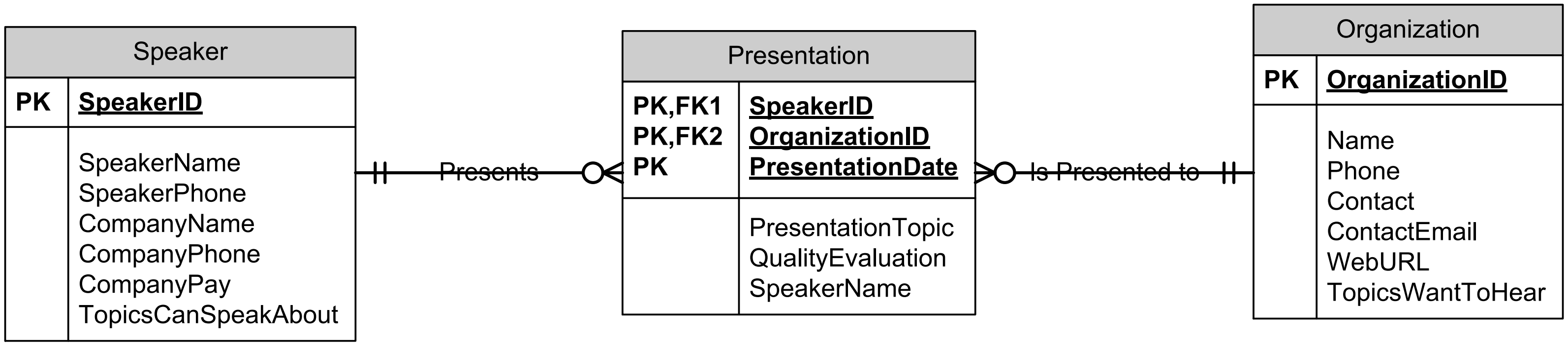
- Unnormalized data is essentially one or a couple of entities.

- Unnormalized data usually has significant data redundancy.

Steps in Normalization

- There are three basic “problems” that a designer identifies with a data model:
  1. Multi-valued attributes (also called “repeating groups” of attributes).
  2. Partial functional dependencies.
  3. Transitive dependencies.
- For each problem, the designer takes the “problem” data and puts it into a separate entity.
- The designer creates an appropriate primary key and relates the new entity to the existing entity with a foreign key.

Exercise example – speaker database



Speaker	
PK	**SpeakerID**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay TopicsCanSpeakAbout

Presents

Presentation	
PK,FK1 PK,FK2 PK	**SpeakerID** **OrganizationID** **PresentationDate**
	PresentationTopic QualityEvaluation SpeakerName

Is Presented to

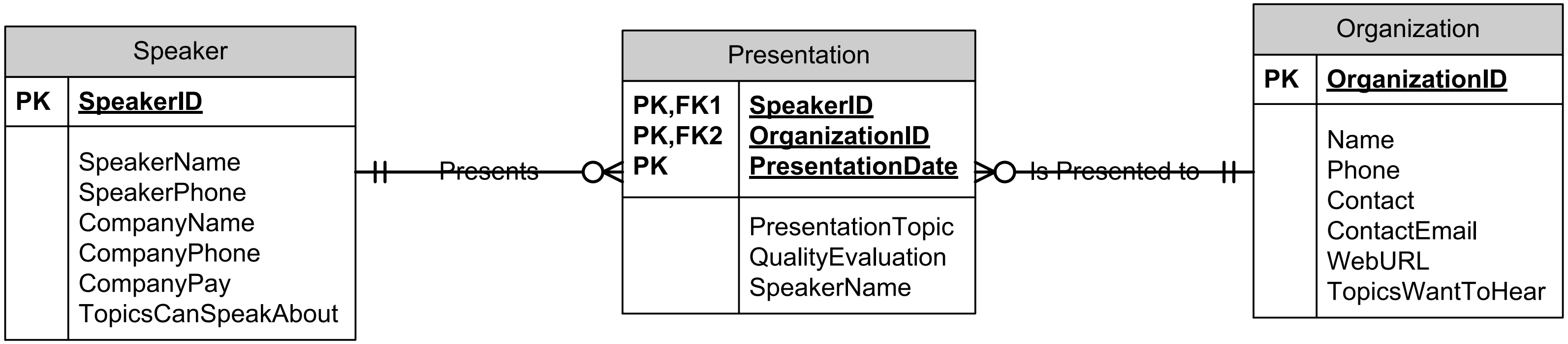
Organization	
PK	**OrganizationID**
	Name Phone Contact ContactEmail WebURL TopicsWantToHear

The application is a database to keep track of speakers giving presentations for student organizations on campus. Student organizations, such as Beta Alpha Psi, Business Student Council, Cyber Security Center, Economics Club, etc. have meetings where they bring in speakers from the business community to give presentations about various topics. The student organization leaders decided it would be helpful to have a centralized database of all presentations given to all student organizations so that they wouldn't duplicate speakers or topics too often. They also thought it would be great to have a way to find the names of potential speakers for future meetings.

Speaker ID	Speaker Name	Speaker Phone	Company Name	Company Phone	Company Pay	Topics to present
12	Beth Adams	775-233-4444	SNC	775-652-8991	No	• ERP Systems • Social Media Marketing • FinTech: Cyber threats

Organization ID	Name	Contact	Contact Email	WebURL	Topics to hear
3	Business Student Council	James Doe	doe@unr.edu	www.dgs.org	• Career Skills • Social Media Marketing • FinTech • AI & Decision making

Speaker ID	Organization ID	Presentation Date	Presentation Topic	Quality Evaluation	Speaker Name
12	3	2/24/2025	Social Media Marketing	Clear discussion but not very dynamic or energetic. Good examples, but boring.	Beth Adams



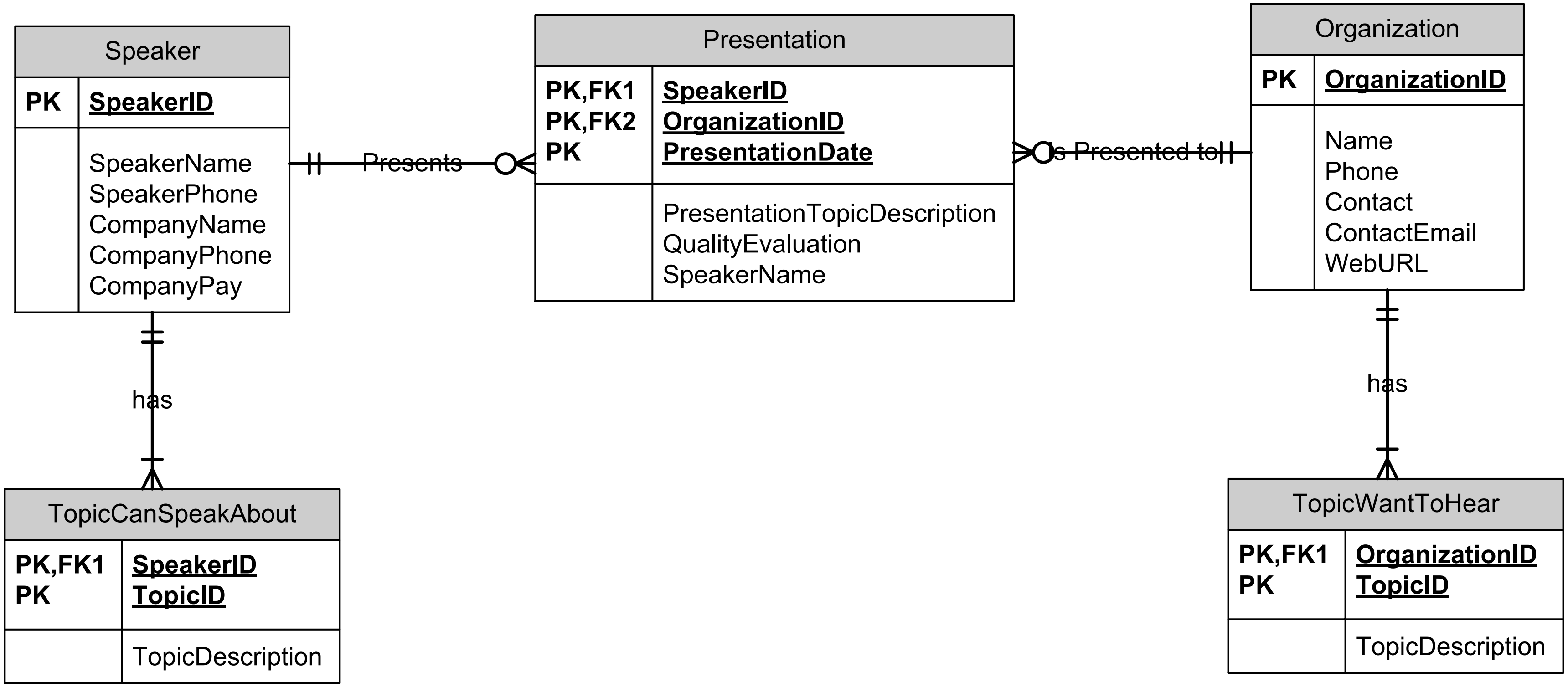
Speaker	
PK	**SpeakerID**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay TopicsCanSpeakAbout

Presentation	
PK,FK1 **PK,FK2** **PK**	**SpeakerID** **OrganizationID** **PresentationDate**
	PresentationTopic QualityEvaluation SpeakerName

Organization	
PK	**OrganizationID**
	Name Phone Contact ContactEmail WebURL TopicsWantToHear

First Normal Form

- First normal form: Remove multi-valued attributes.
  - A multi-valued attribute is an attribute or group of attributes that can have more than one value for an instance of an entity. If it is a group of attributes, it is called a “repeating group.”
- To see whether a data model is in first normal form:
  - Identify repeating groups/multi-valued attributes and place them as separate entities in the model.
- How to fix the problem:
  - Identify a primary key for the new entity. The primary key will most likely be concatenated.
  - Create the relationships between entities.
  - Divide m:n relationships with appropriate intersection entities.



Speaker	
PK	**SpeakerID**
	SpeakerName SpeakerPhone CompanyName CompanyPhone CompanyPay

Presentation	
PK,FK1 PK,FK2 PK	**SpeakerID** **OrganizationID** **PresentationDate**
	PresentationTopicDescription QualityEvaluation SpeakerName

Organization	
PK	**OrganizationID**
	Name Phone Contact ContactEmail WebURL

TopicCanSpeakAbout	
PK,FK1 PK	**SpeakerID** **TopicID**
	TopicDescription

TopicWantToHear	
PK,FK1 PK	**OrganizationID** **TopicID**
	TopicDescription

SpeakerID -> SpeakerName, SpeakerPhone, CompanyName, CompanyPhone, CompanyPay

SpeakerID, OrganizationID, PresentationDate -> PresentationTopic, QualityEvaluation, SpeakerName

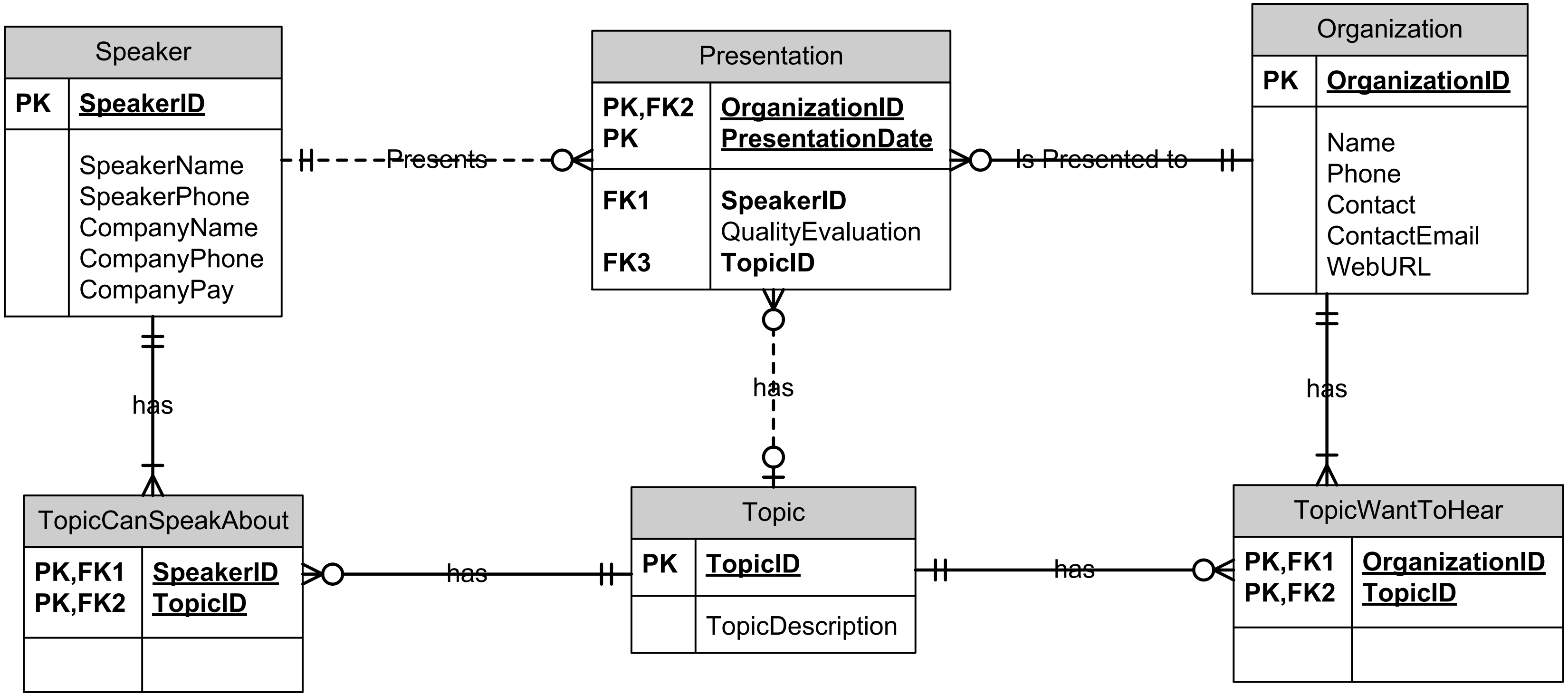
OrganizationID -> Name, Phone, Contact, ContactEmail, WebURL

SpeakerID, TopicID -> TopicDescription

OrganizationID, TopicID -> TopicDescription

Second Normal Form

- Second normal form: Remove partial functional dependencies.
- A partial functional dependency is a situation in which one or more non-key attributes are functionally dependent on part, but not all, of the primary key.
  - Partial functional dependencies occur only with concatenated keys.
- Examples of partial functional dependencies:
  - PatientID, TreatmentDateTime → PatName, TstResults, TrtID, LocID
  - CourseID, StudentID → CourseTitle, Grade
- Which entities on the current ERD have a concatenated primary key?



SpeakerID -> SpeakerName, SpeakerPhone, CompanyName, CompanyPhone, CompanyPay

OrganizationID, PresentationDate -> SpeakerID, QualityEvaluation, TopicID

OrganizationID -> Name, Phone, Contact, ContactEmail, WebURL

TopicID -> TopicDescription

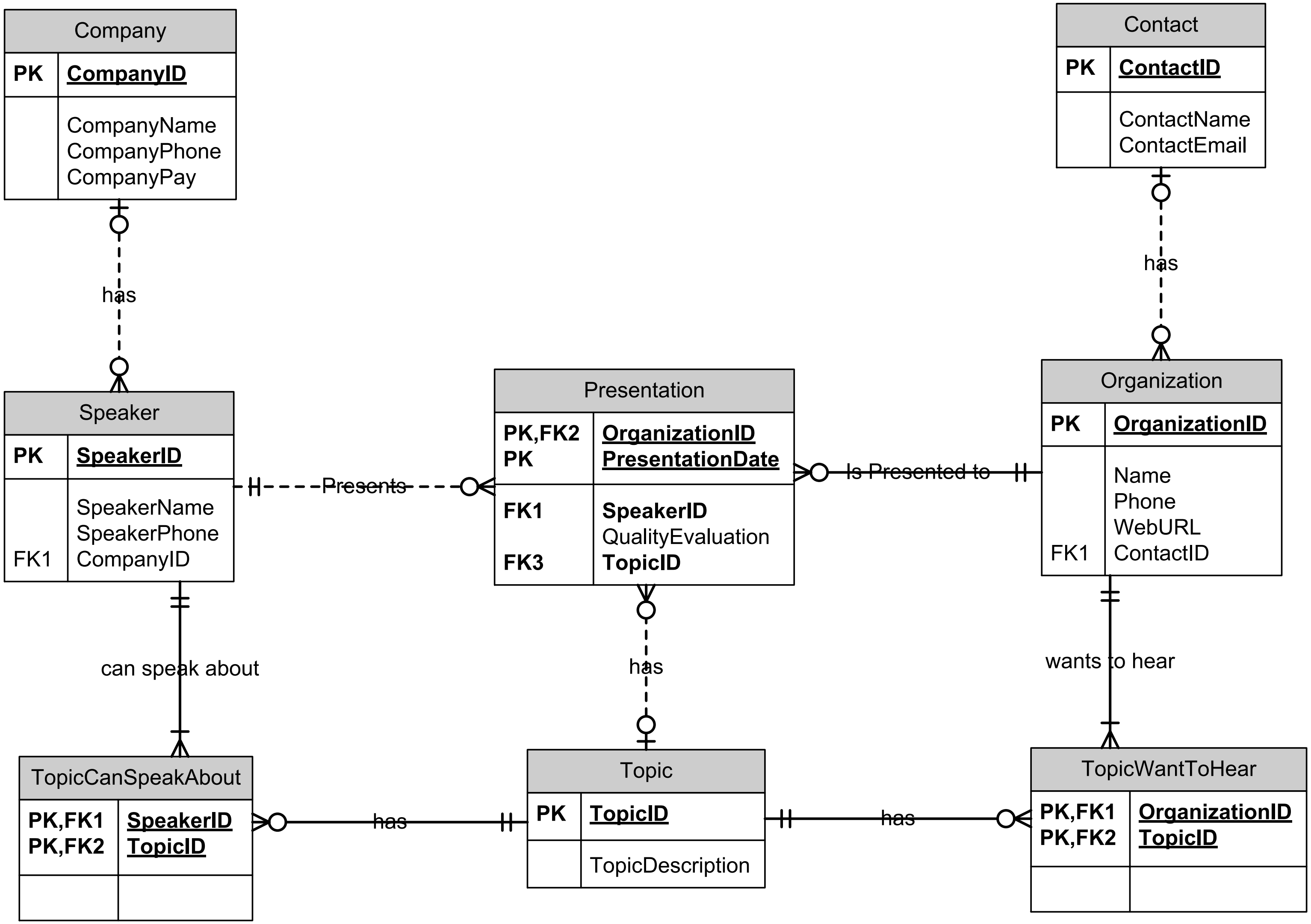
SpeakerID, TopicID ->

OrganizationID, TopicID ->

Third normal form

- Third normal form: Remove transitive dependencies.
    - A transitive dependency occurs when a non-key attribute is functionally dependent on one or more non-key attributes.
- Third normal form examines entities with single primary keys and removes the “floating” or transitive dependencies.
- It may be possible to have attributes that are determined by other attributes, rather than by the primary key. They must be removed into entities with appropriate primary keys.
- Example of transitive dependency:

??->CompanyName, CompanyPhone, CompanyPay



Third Normal Form

Functional Dependency Diagrams

CompanyID -> CompanyName, CompanyPhone, CompanyPay

SpeakerID -> SpeakerName, SpeakerPhone, CompanyID

ContactID -> ContactName, ContactEmail

OrganizationID, PresentationDate -> SpeakerID, QualityEvaluation, TopicID

OrganizationID -> Name, Phone, ContactID, WebURL

TopicID -> TopicDescription

SpeakerID, TopicID ->

OrganizationID, TopicID ->

Summary of normalization process

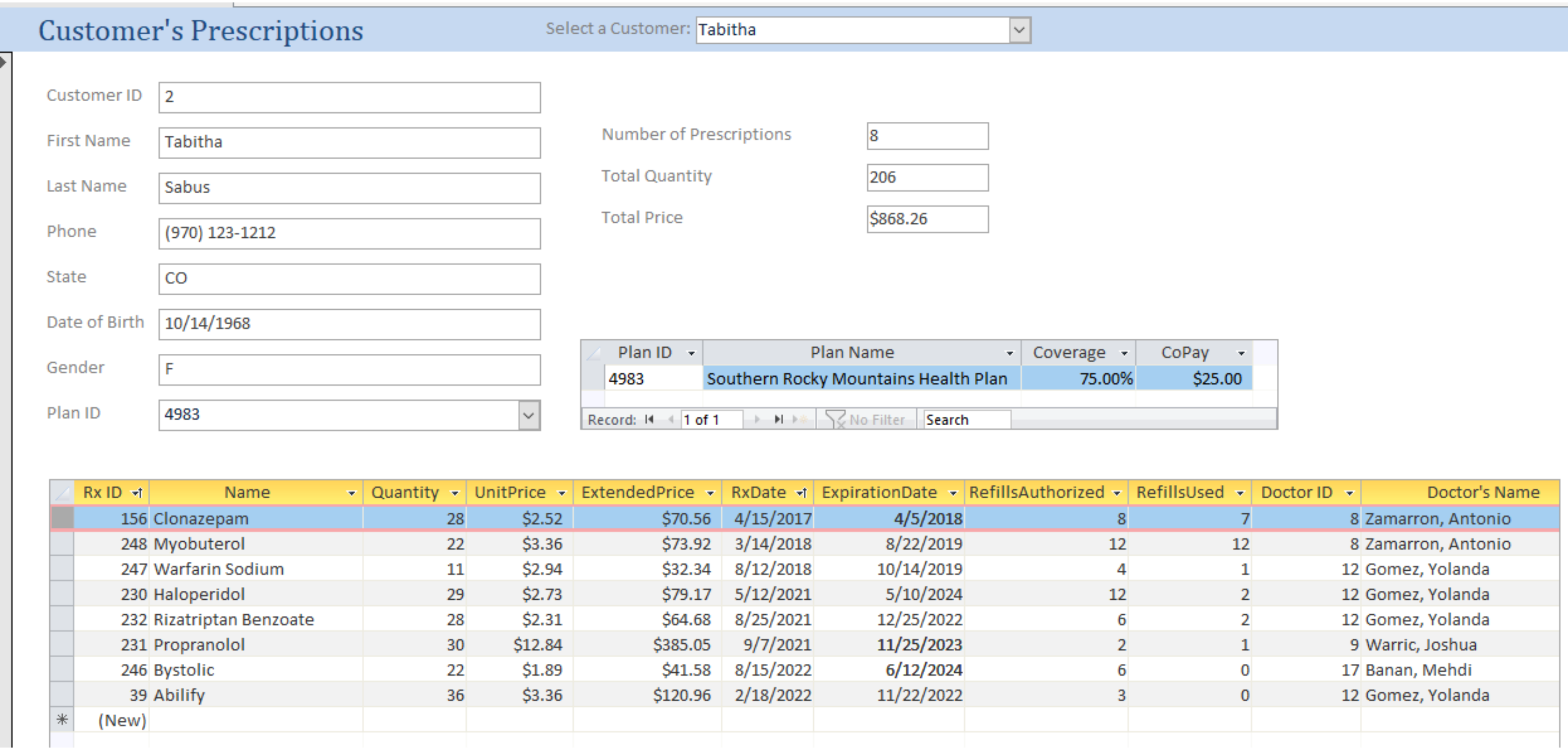
- Examine and evaluate the logical data model:
    - Find the **repeating groups and/or multi-valued attributes** and put the model into first normal form. Identify primary key fields for any new entities. Create concatenated keys as necessary. Relate entities with foreign keys.

    - Find the **functional dependencies**. Identify the **partial functional dependencies** and put the model into second normal form. Identify primary key fields for any new entities. Create concatenated keys as necessary. Relate entities with foreign keys.

    - Find the **transitive dependencies** and put the model into third normal form. Identify primary key fields for any new entities. Relate entities with foreign keys.

The goals of this exercise are to evaluate and fix an ERD based on the principles of normalization.

- First, examine the ERD and identify any multi-valued attributes. Ask yourself if any of the data in the existing entities will have more than one value for a given entity instance. If you see a group of multi-valued attributes (repeating group), put the data that forms the repeating group into its own entity with an applicable primary key. The primary key of the new entity will most likely be concatenated.
- Second, examine the ERD and look only at the entities with concatenated primary keys. Make sure that each attribute within an entity with a concatenated primary key is dependent on the whole primary key, rather than just part of the primary key. For any attribute that is not dependent on the whole primary key, remove that attribute from the entity and put it into an entity where it is dependent on the whole primary key. Make a new entity if necessary. The primary key of the new entity you create may or may not be concatenated.
- Third, examine the ERD and look at each entity. Check to make sure that each and every attribute within the entity is dependent on the primary key, and not some other attribute within the entity or the primary key of another entity. If such a “transitive dependency” occurs, then remove that attribute from the entity and put it in another entity where it is dependent on the primary key. Make a new entity if necessary.



Customer's Prescriptions

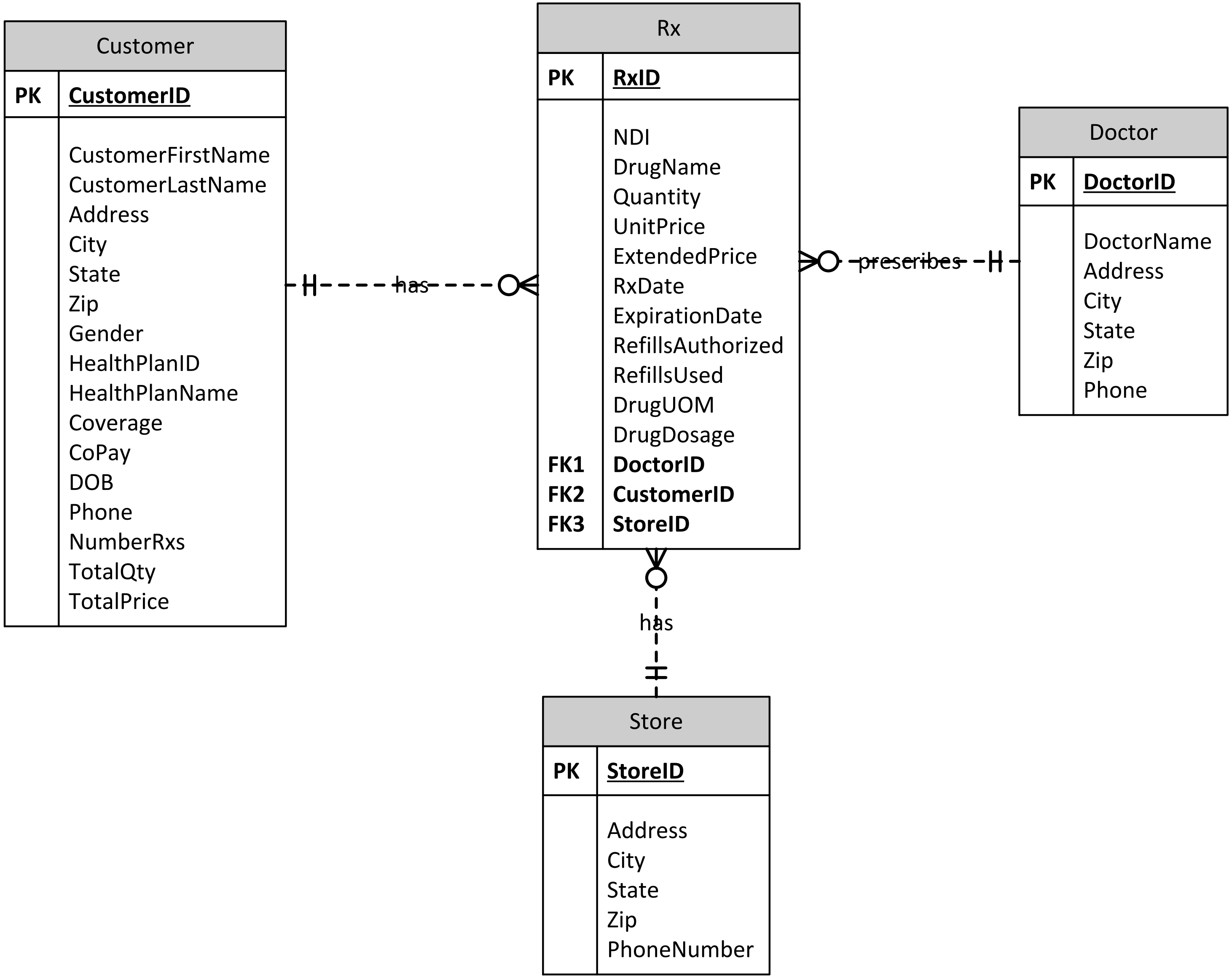
Select a Customer: Tabitha

Customer ID	2
First Name	Tabitha
Last Name	Sabus
Phone	(970) 123-1212
State	CO
Date of Birth	10/14/1968
Gender	F
Plan ID	4983

Number of Prescriptions	8
Total Quantity	206
Total Price	$868.26

Plan ID	Plan Name	Coverage	CoPay
4983	Southern Rocky Mountains Health Plan	75.00%	$25.00

Rx ID	Name	Quantity	UnitPrice	ExtendedPrice	RxDate	ExpirationDate	RefillsAuthorized	RefillsUsed	Doctor ID	Doctor's Name
156	Clonazepam	28	$2.52	$70.56	4/15/2017	**4/5/2018**	8	7	8	Zamarron, Antonio
248	Myobuterol	22	$3.36	$73.92	3/14/2018	8/22/2019	12	12	8	Zamarron, Antonio
247	Warfarin Sodium	11	$2.94	$32.34	8/12/2018	10/14/2019	4	1	12	Gomez, Yolanda
230	Haloperidol	29	$2.73	$79.17	5/12/2021	5/10/2024	12	2	12	Gomez, Yolanda
232	Rizatriptan Benzoate	28	$2.31	$64.68	8/25/2021	12/25/2022	6	2	12	Gomez, Yolanda
231	Propranolol	30	$12.84	$385.05	9/7/2021	**11/25/2023**	2	1	9	Warric, Joshua
246	Bystolic	22	$1.89	$41.58	8/15/2022	**6/12/2024**	6	0	17	Banan, Mehdi
39	Abilify	36	$3.36	$120.96	2/18/2022	11/22/2022	3	0	12	Gomez, Yolanda
(New)										



Customer	
PK	**CustomerID**
	CustomerFirstName CustomerLastName Address City State Zip Gender HealthPlanID HealthPlanName Coverage CoPay DOB Phone NumberRxs TotalQty TotalPrice

Rx	
PK	**RxID**
	NDI DrugName Quantity UnitPrice ExtendedPrice RxDate ExpirationDate RefillsAuthorized RefillsUsed DrugUOM DrugDosage
FK1	**DoctorID**
FK2	**CustomerID**
FK3	**StoreID**

Doctor	
PK	**DoctorID**
	DoctorName Address City State Zip Phone

Store	
PK	**StoreID**
	Address City State Zip PhoneNumber