IT-234 – database concepts

UNIT 2 – THE LOGICAL DATABASE MODEL AND DATABASE NORMALIZATION

overview

Logical database design typically entails synthesizing individual data elements into normalized tables after careful analysis of data element interdependencies defined by business requirements analysis.

The logical data model specifically adds attributes, primary keys, and foreign keys.

overview

The tables will be able to store data about the organization's entities in a non-redundant manner, and foreign keys will be placed in the tables so that all the relationships among the entities will be supported.

You are adding more detail to your data model, but there is still no dependence on any database system.

overview

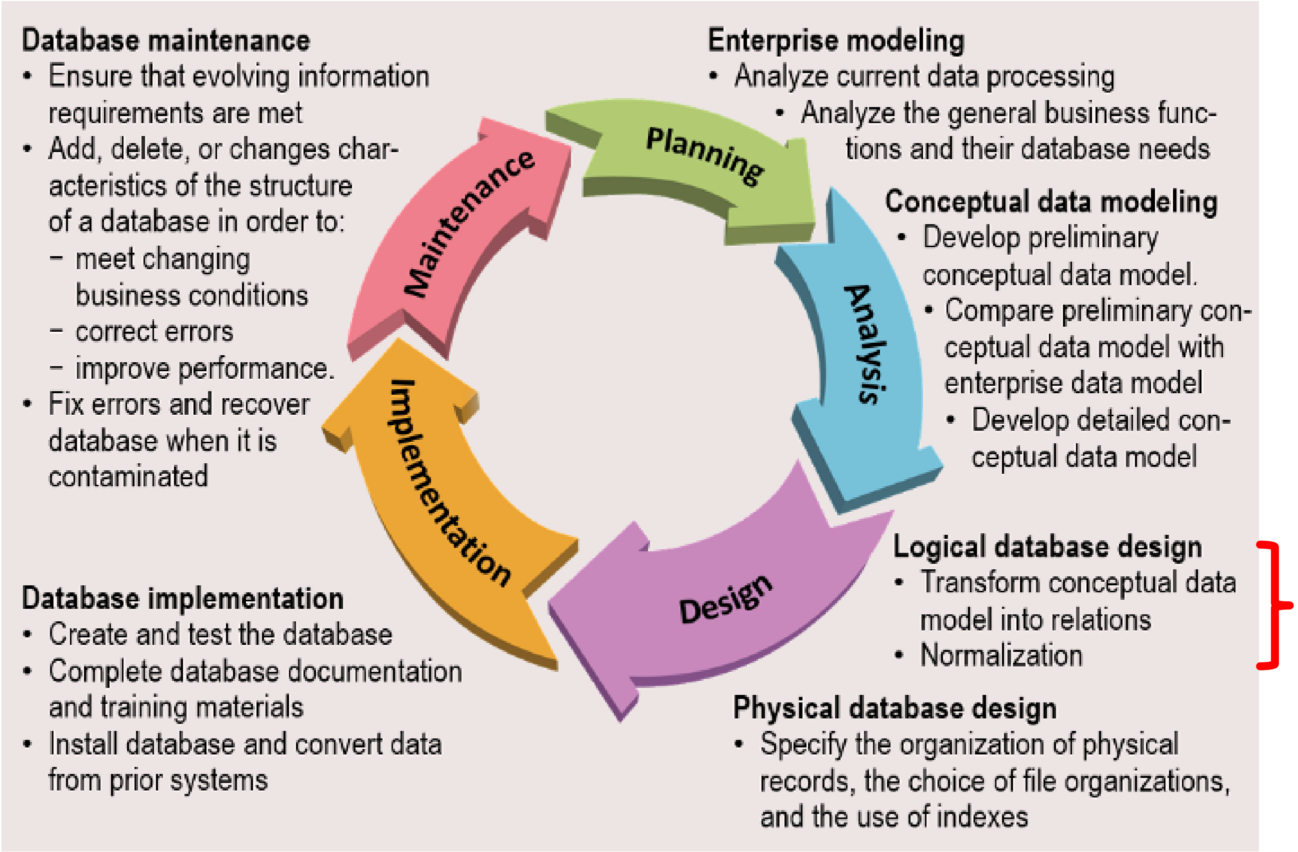
The logical data model will be describing the data requirements from a business point of view.

The goal of logical database design is to create well-structured relations that properly reflect the company's business environment.

overview

After completing this unit, you should be able to:

- Identify attributes for entities in the database.
- Define data type and nullability.
- Identify all primary keys for entities in the database.
- Recognize any foreign keys required for entities in the database.
- Create an entity relationship diagram (ERD) that reflects the logical data model.



Database life cycle

Components of THE relational model

Data structure

- Tables (relations), rows, columns

Data manipulation

- Powerful SQL operations for retrieving and modifying data

Data integrity

- Mechanisms for implementing business rules that maintain integrity of manipulated data

RelationS

A relation is a named, two-dimensional table of data.

A table consists of rows (records) and columns (attribute or field).

Requirements for a table to qualify as a relation:

- It must have a unique name.
- Every attribute value must be atomic (not multivalued, not composite).
- Every row must be unique (can't have two rows with exactly the same values for all their fields).
- Attributes (columns) in tables must have unique names.

NOTE: All *relations* are in ***1st Normal form.***

RELATIONS

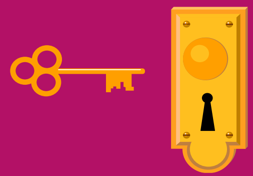
Relations (tables) correspond with entity types and with many-to-many relationship types.

Rows correspond with entity instances and with many-to-many relationship instances.

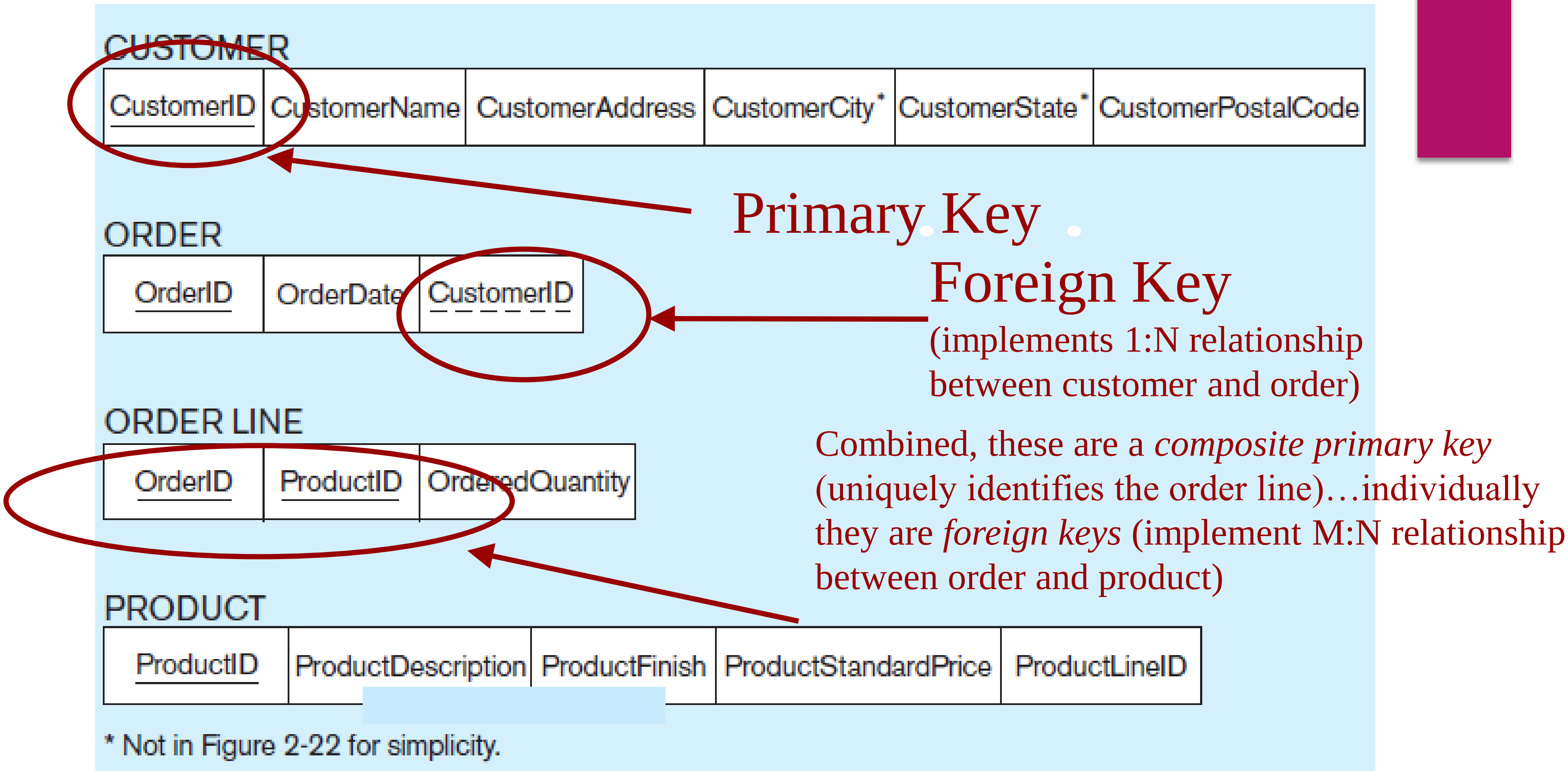
Columns correspond with attributes.

NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model).

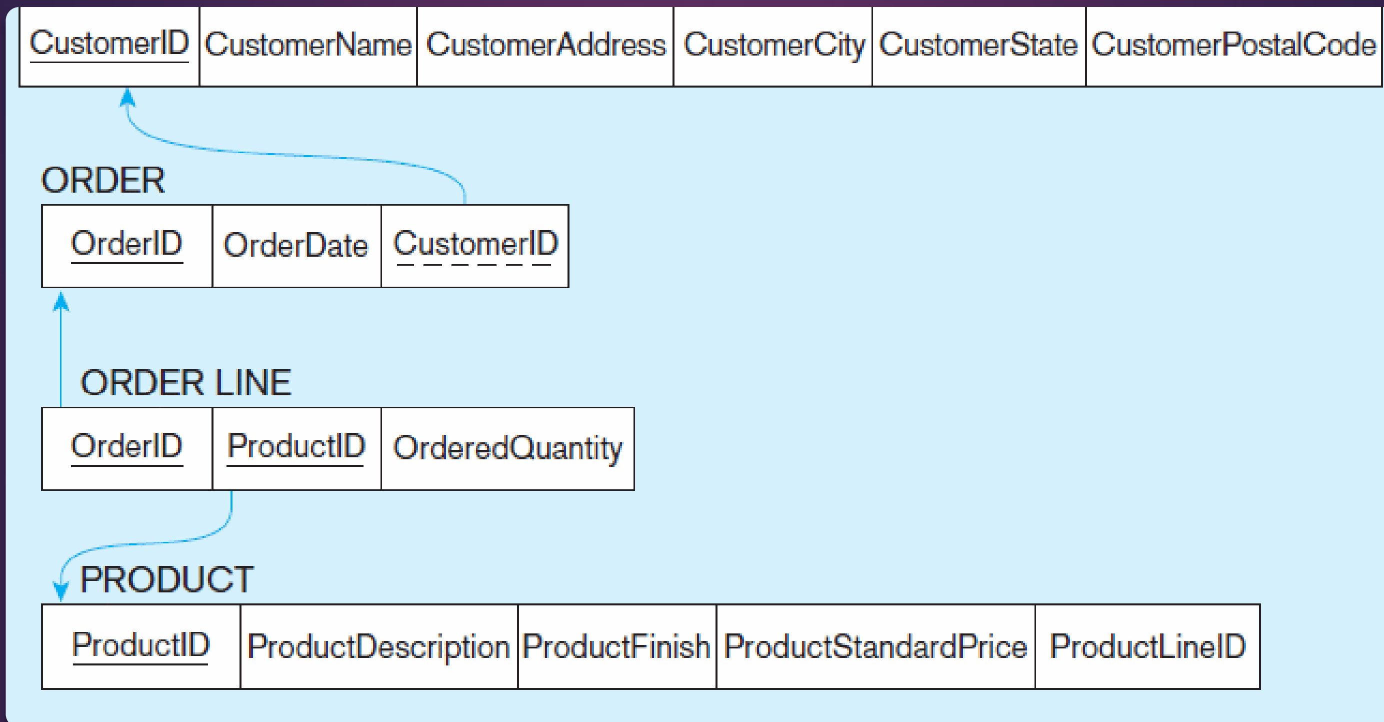
Key Fields



- Keys are special fields that serve two main purposes:
  - ***Primary keys*** are unique identifiers of the relation. Examples include employee numbers, social security numbers, etc. *This guarantees that all rows are unique.*
  - ***Foreign keys*** are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship).
- Keys can be ***simple*** (a single field) or ***composite*** (more than one field).
- Keys usually are used as indexes to speed up the response to user queries



Schema for four relations (Pine Valley Furniture Company)



Referential integrity constraints (Pine Valley Furniture)

REFERENTIAL INTEGRITY CONSTRAINTS ARE DRAWN VIA ARROWS FROM DEPENDENT TO PARENT TABLE

```
CREATE TABLE Customer_T
        (CustomerID                 NUMBER(11,0)    NOT NULL,
        CustomerName                VARCHAR2(25)    NOT NULL,
        CustomerAddress             VARCHAR2(30),
        CustomerCity                VARCHAR2(20),
        CustomerState               CHAR(2),
        CustomerPostalCode          VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                    NUMBER(11,0)    NOT NULL,
        OrderDate                   DATE DEFAULT SYSDATE,
        CustomerID                  NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));

CREATE TABLE Product_T
        (ProductID                  NUMBER(11,0)    NOT NULL,
        ProductDescription          VARCHAR2(50),
        ProductFinish               VARCHAR2(20),
        ProductStandardPrice        DECIMAL(6,2),
        ProductLineID               NUMBER(11,0),
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
        (OrderID                    NUMBER(11,0)    NOT NULL,
        ProductID                   NUMBER(11,0)    NOT NULL,
        OrderedQuantity             NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID));
```

Referential integrity constraints are implemented with foreign key to primary key references

SQL table definitions

Transforming ER Diagrams into Relations

Mapping Regular Entities to Relations

- Simple attributes: E-R attributes map directly onto the relation
- Composite attributes: Use only their simple, component attributes
- Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity

CUSTOMER ENTITY TYPE WITH SIMPLE ATTRIBUTES

CUSTOMER

Customer ID
Customer Name
Customer Address
Customer Postal Code

CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

CUSTOMER relation

Mapping a Regular Entity

CUSTOMER entity type with composite attribute

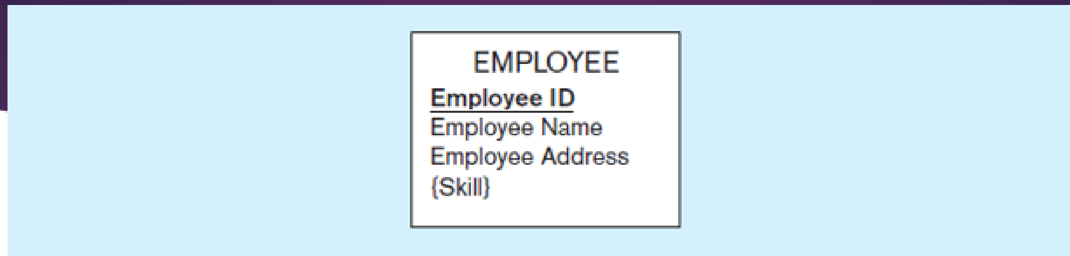
CUSTOMER
<u>Customer ID</u>
Customer Name
Customer Address
(CustomerStreet, CustomerCity, CustomerState)
Customer Postal Code

CUSTOMER relation with address detail

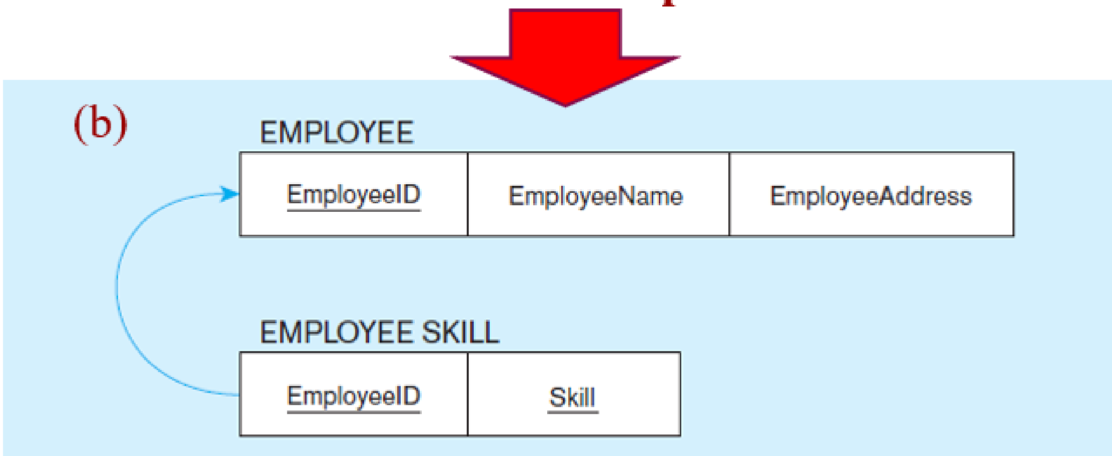
CUSTOMER

<u>CustomerID</u>	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

Mapping a Composite Attribute



Multivalued attribute becomes a separate relation with foreign key



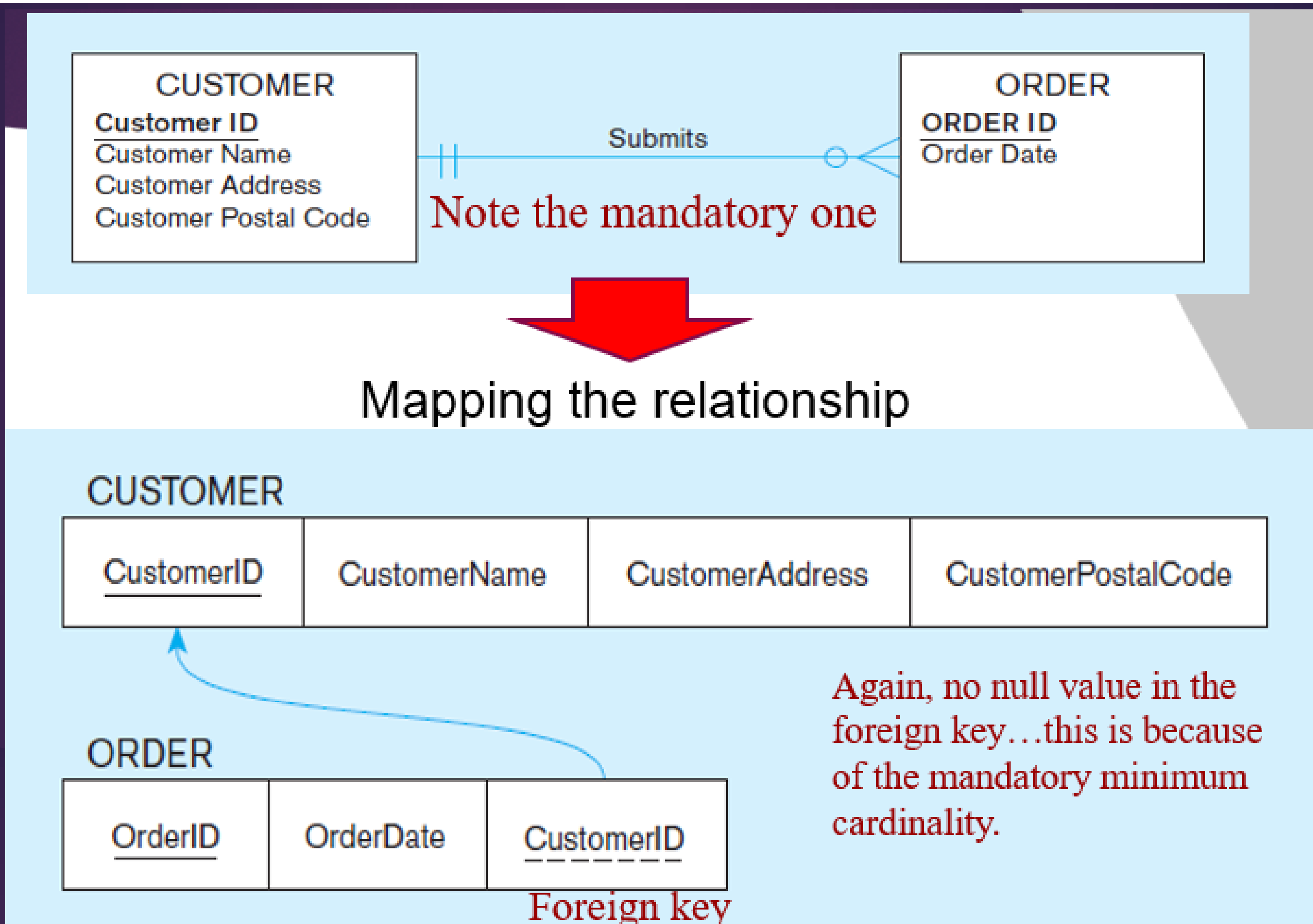
One–to–many relationship between original entity and new relation

Mapping an Entity with a Multivalued Attribute

Transforming ER Diagrams into Relations (cont.)

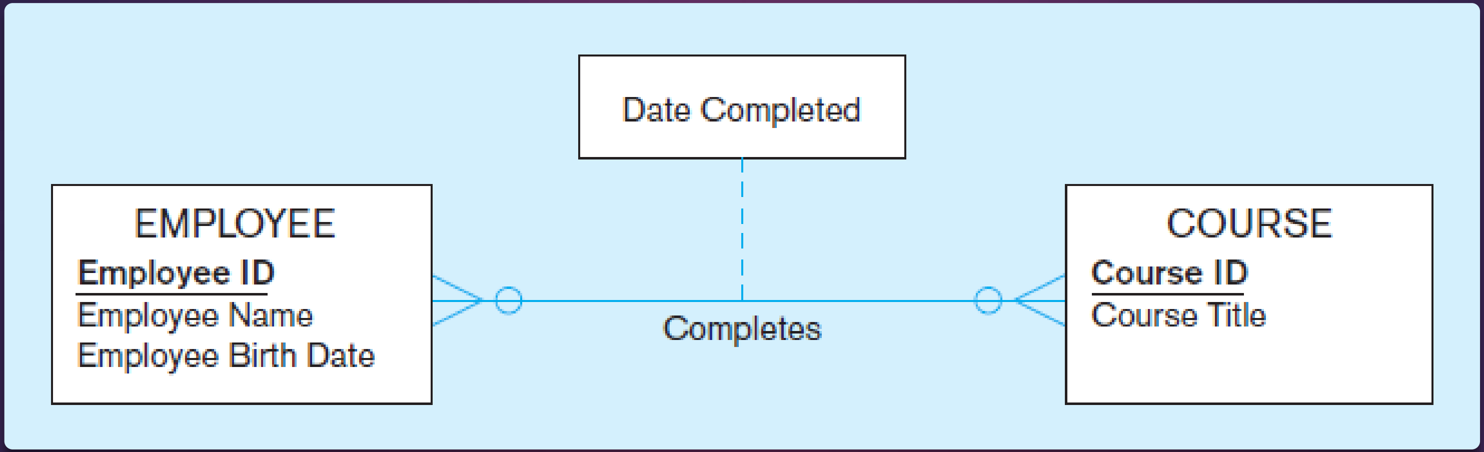
Mapping Binary Relationships

- **One-to-Many**–Primary key on the one side becomes a foreign key on the many side
- **Many-to-Many**–Create a ***new relation*** with the primary keys of the two entities as its primary key
- **One-to-One**–Primary key on mandatory side becomes a foreign key on optional side



Example of Mapping a 1:M Relationship

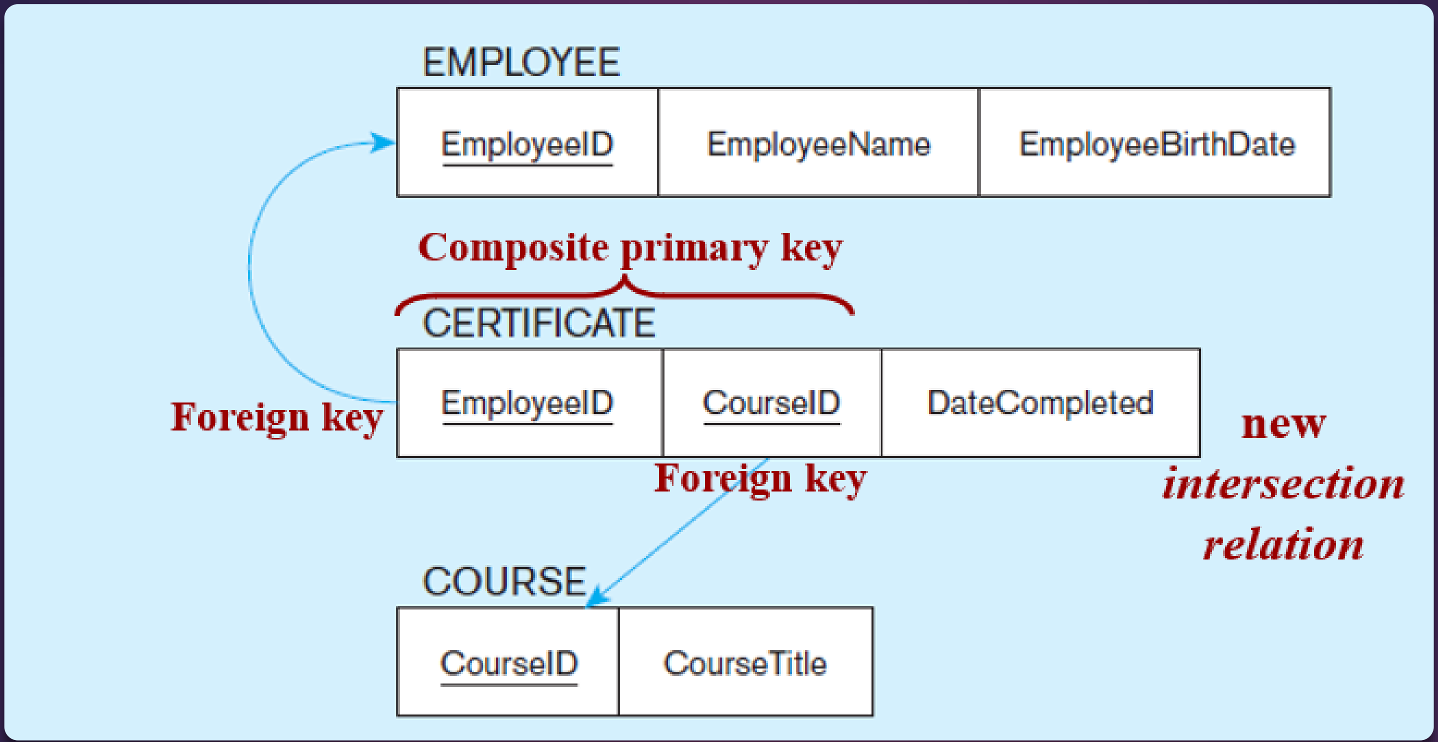
RELATIONSHIP BETWEEN CUSTOMERS AND ORDERS



The *Completes* relationship will need to become a separate relation.

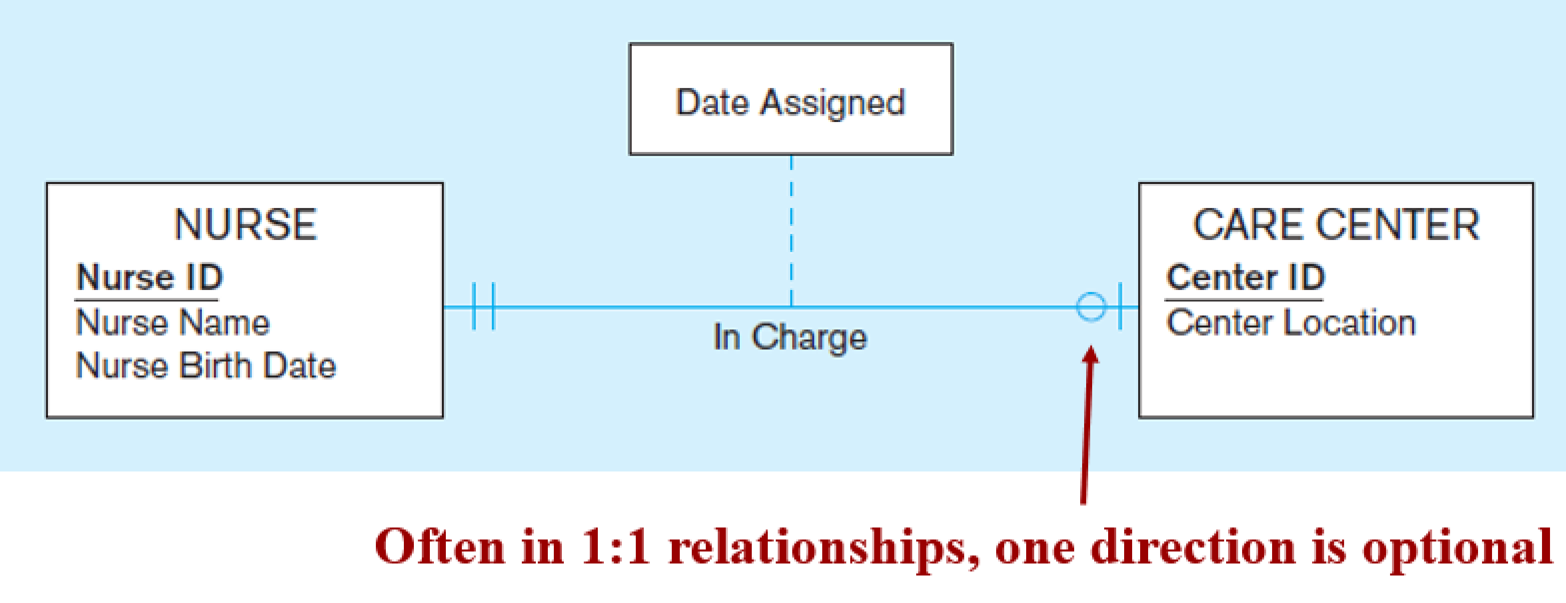
Example of Mapping an M:N Relationship

“COMPLETES” RELATIONSHIP (M:N)



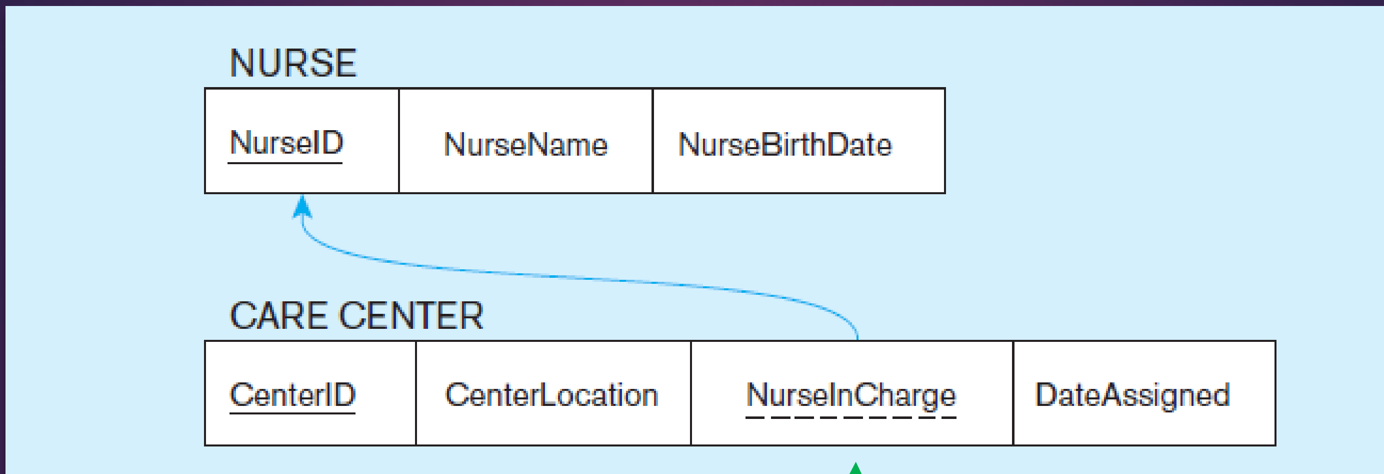
Example of Mapping an M:N Relationship (cont.)

THREE RESULTING RELATIONS



Example of Mapping a Binary 1:1 Relationship

"IN CHARGE" RELATIONSHIP (BINARY 1:1)



Foreign key goes in the relation on the optional side, matching the primary key on the mandatory side

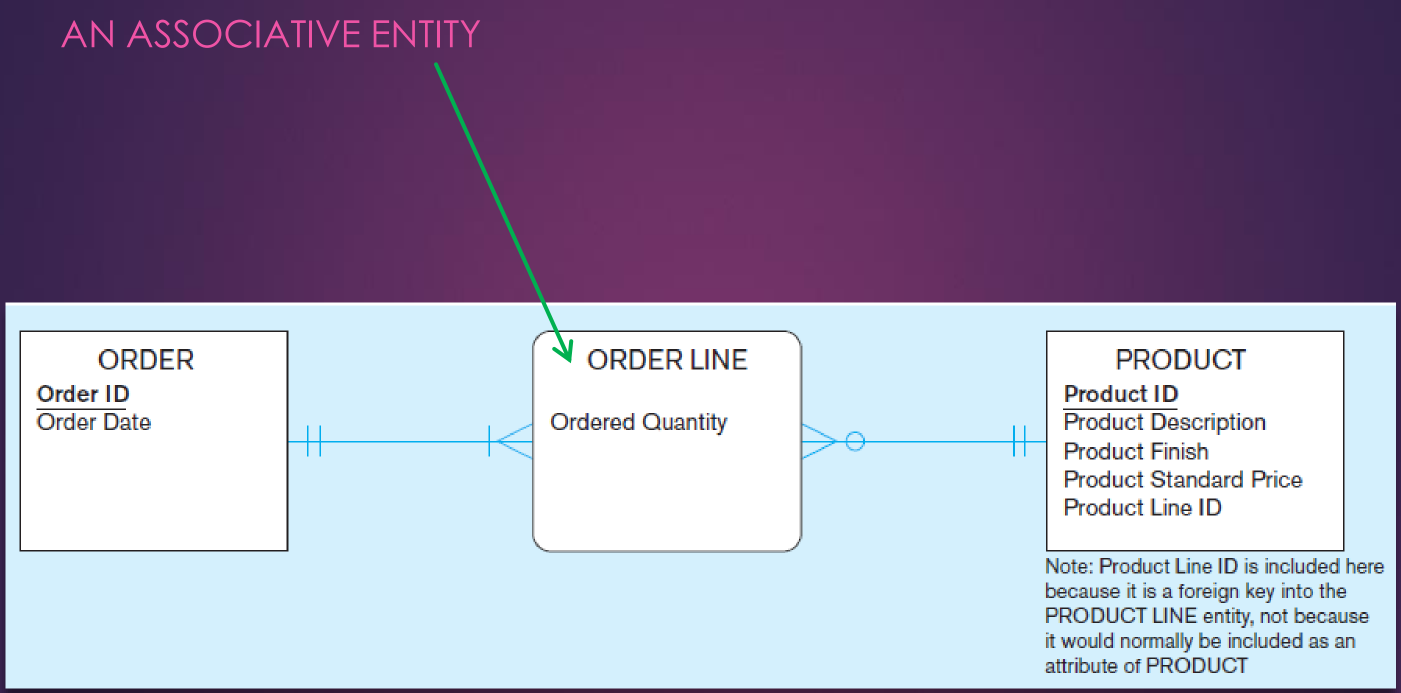
Example of Mapping a Binary 1:1 Relationship (cont.)

RESULTING RELATIONS

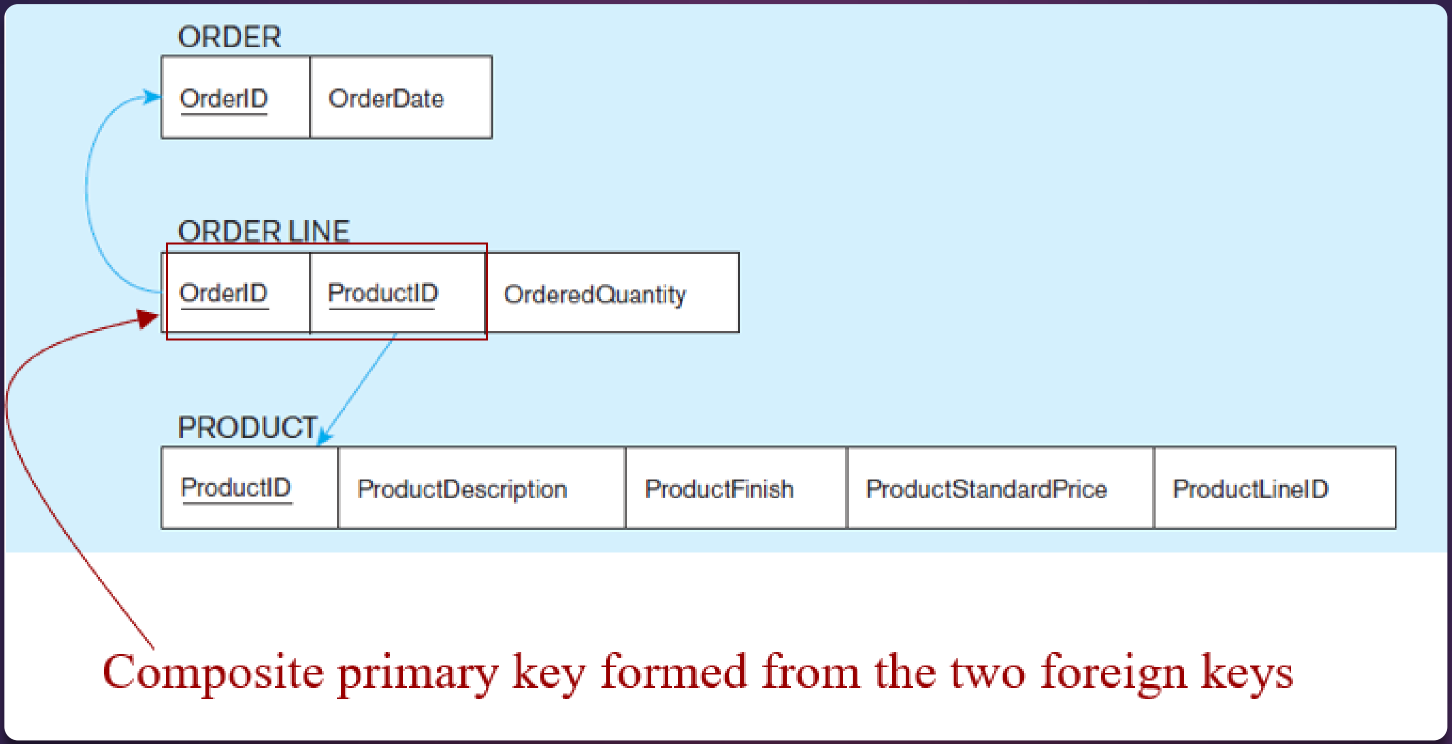
Transforming ER Diagrams into Relations (cont.)

Mapping Associative Entities

- Identifier Not Assigned
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)
- Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

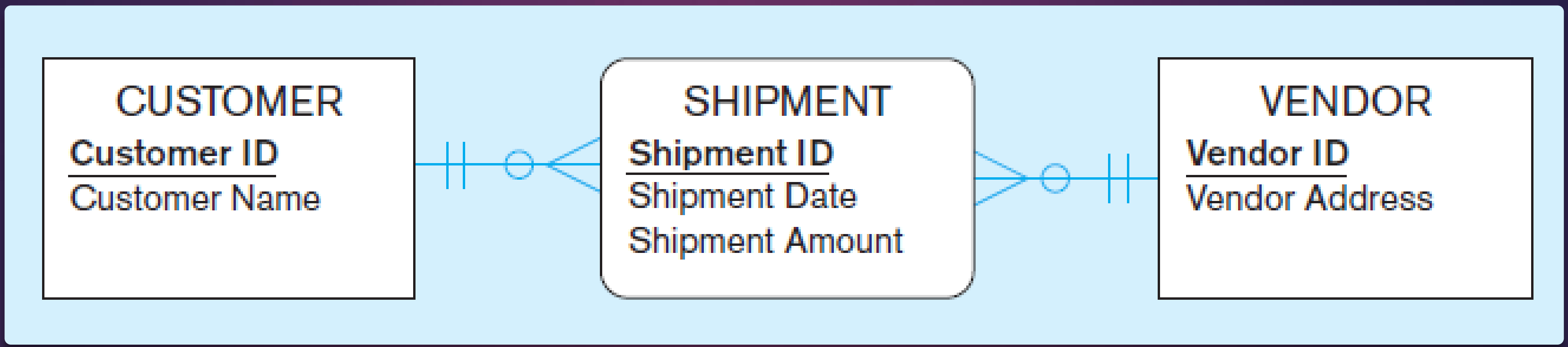


Example of Mapping an Associative Entity



Example of Mapping an Associative Entity (cont.)

THREE RESULTING RELATIONS

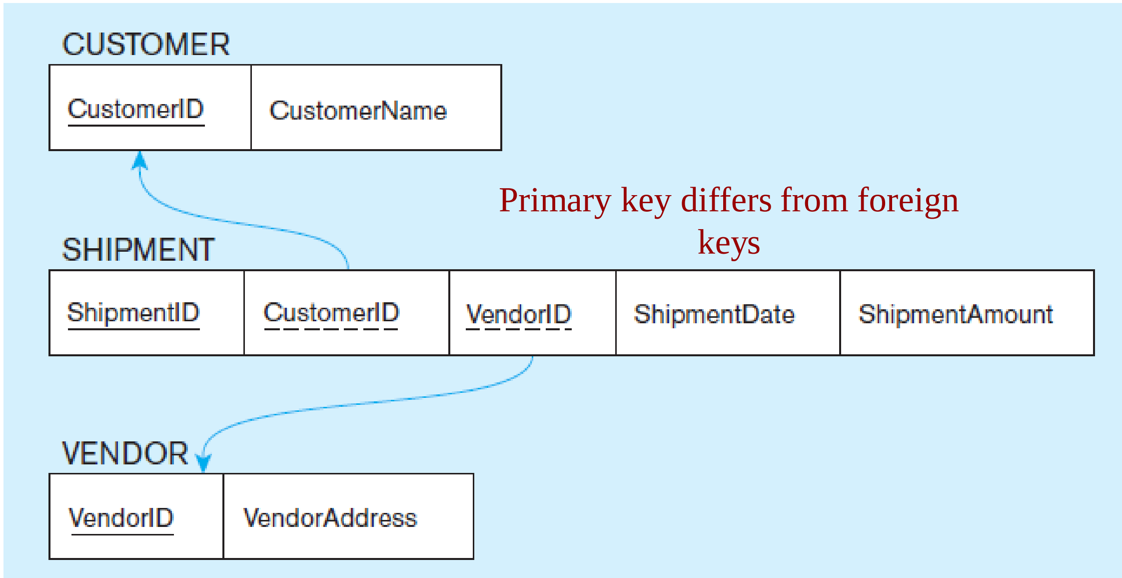


Example of Mapping an Associative Entity with an Identifier

SHIPMENT ASSOCIATIVE ENTITY

Example of Mapping an Associative Entity with an Identifier (cont.)

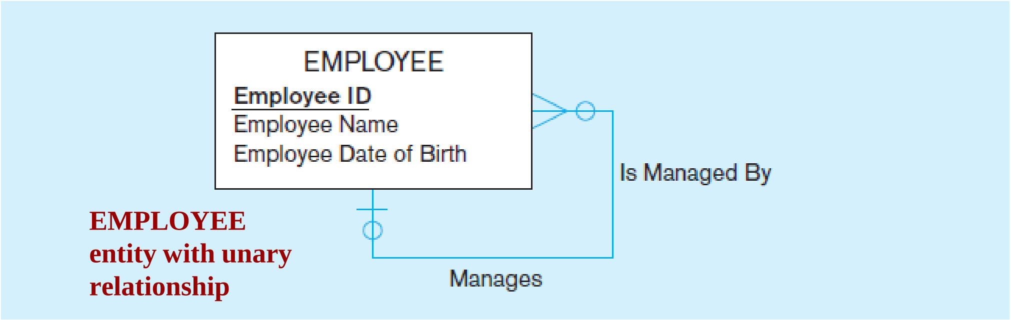
Three Resulting Relations



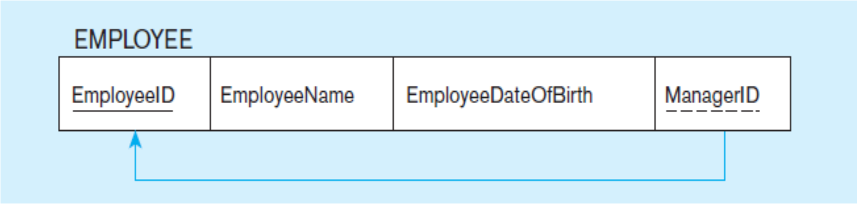
Transforming ER Diagrams into Relations (cont.)

Mapping Unary Relationships

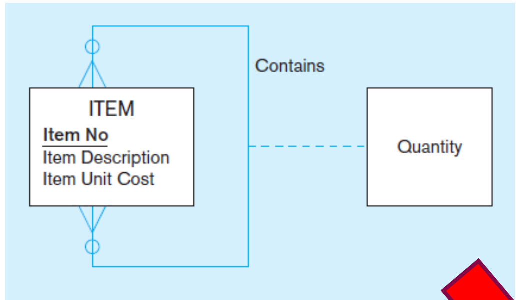
- One-to-Many – Recursive foreign key in the same relation
- Many-to-Many – Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity



EMPLOYEE relation with recursive foreign key

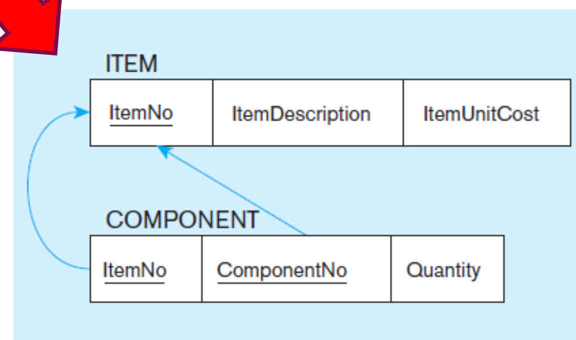


Mapping a Unary 1:N Relationship



Bill-of-materials relationships (unary M:N)

ITEM and COMPONENT relations

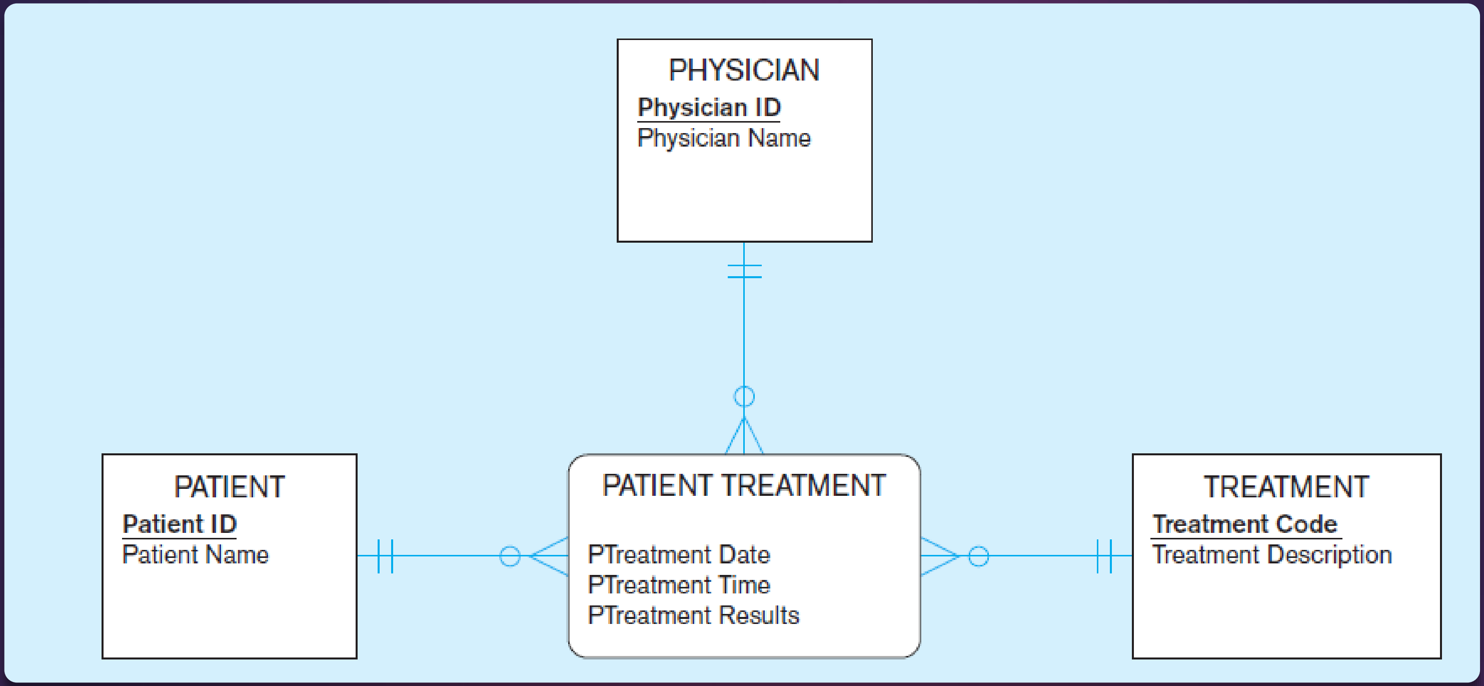


Mapping a unary M:N relationship

Transforming ER Diagrams into Relations (cont.)

Mapping Ternary (and n-ary) Relationships

- One relation for each entity and one for the associative entity
- Associative entity has foreign keys to each entity in the relationship

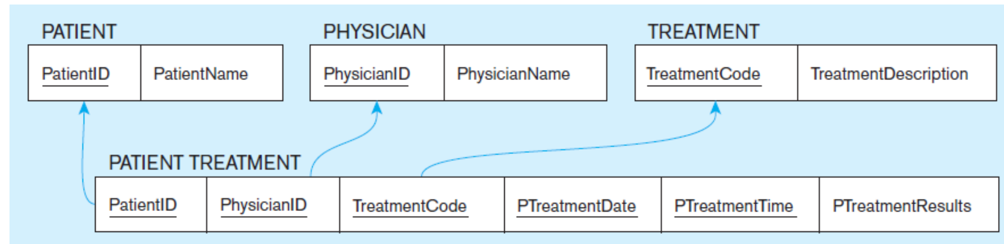


Mapping a Ternary Relationship

PATIENT TREATMENT TERNARY RELATIONSHIP WITH ASSOCIATIVE ENTITY

Mapping a Ternary Relationship (cont.)

Mapping the ternary relationship PATIENT TREATMENT



Remember that the primary key MUST be unique.

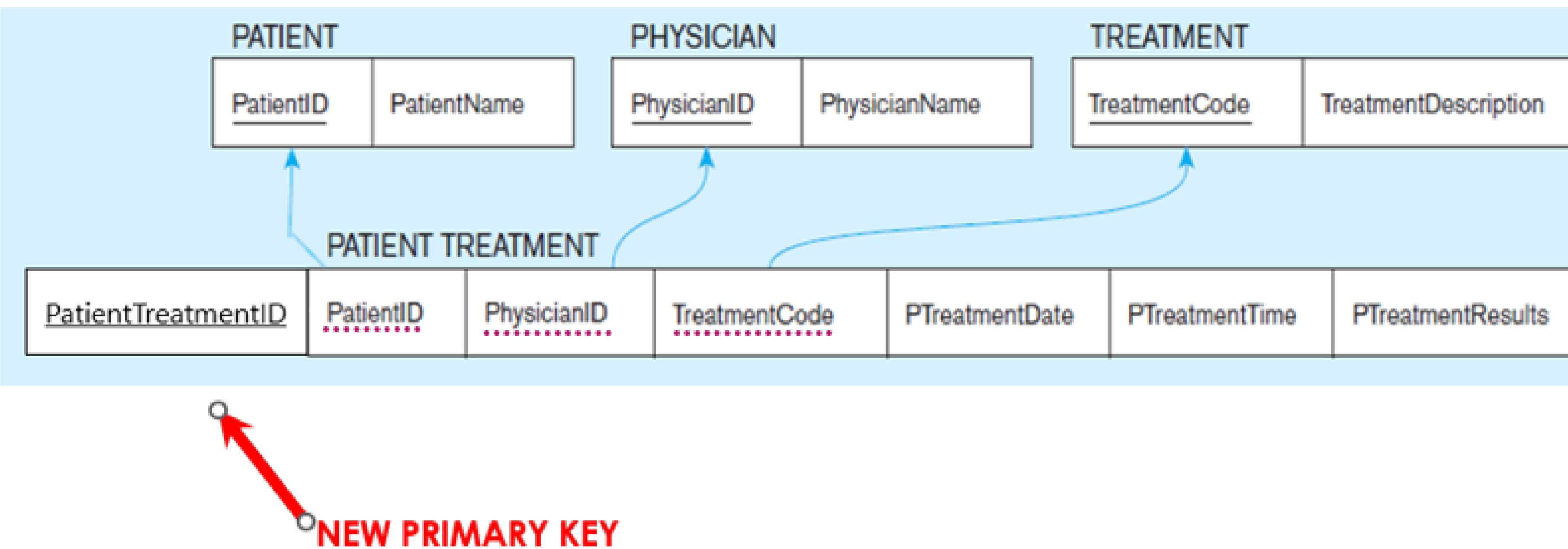
This is why treatment date and time are included in the composite primary key.

But this makes a very cumbersome key…

It would be better to create a surrogate key like Treatment#.

▶Mapping a Ternary Relationship (cont.)

Mapping the ternary relationship PATIENT TREATMENT



Data Normalization

Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***

The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations

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

- **Insertion Anomaly**–adding new rows forces user to create duplicate data
- **Deletion Anomaly**–deleting rows may cause a loss of data that would be needed for other future rows
- **Modification Anomaly**–changing data in a row forces changes to other rows because of duplication

General rule of thumb: A table should not pertain to more than one entity type.

Anomaly Example

EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2015
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/2015
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2015
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2015
110	Chris Lucero	Info Systems	43,000	C++	4/22/2015
190	Lorenzo Davis	Finance	55,000	Tax Acc	
150	Susan Martin	Marketing	42,000	SPSS	6/19/2015
150	Susan Martin	Marketing	42,000	Java	8/12/2015

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

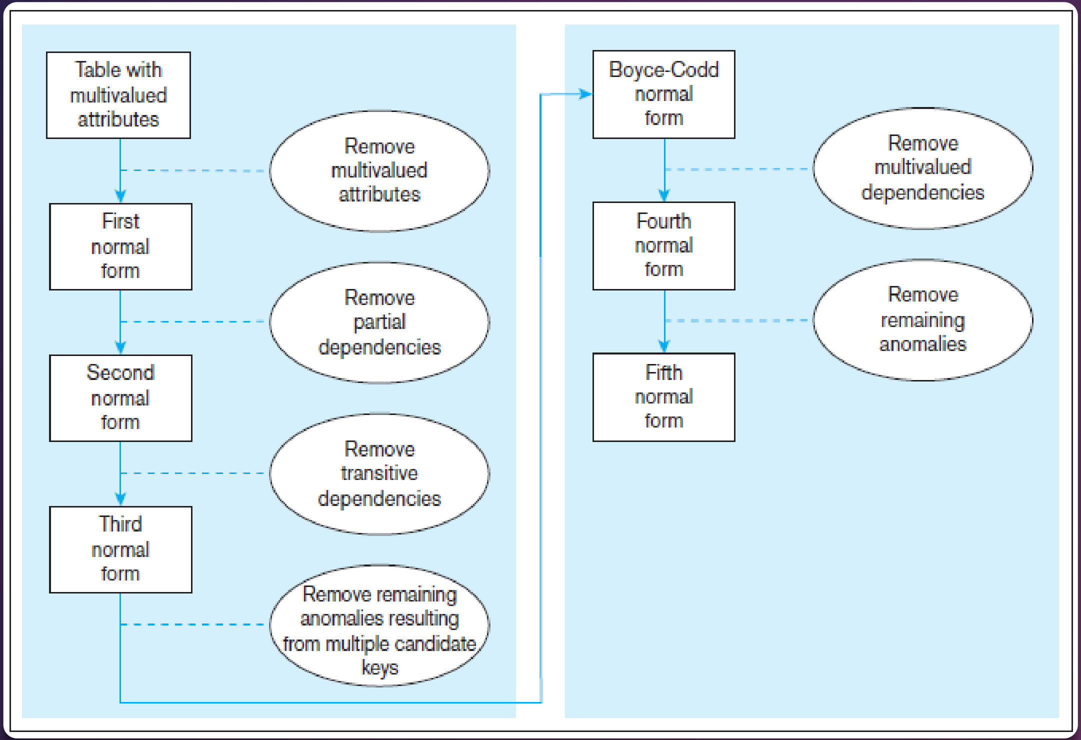
Insertion–can't enter a new employee without having the employee take a class (or at least empty fields of class information)

Deletion–if we remove employee 140, we lose information about the existence of a Tax Acc class

Modification–giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities.



Steps in Normalization

3RD NORMAL FORM IS GENERALLY CONSIDERED SUFFICIENT

Functional Dependencies and Keys

Functional Dependency: The value of one attribute (the ***determinant***) determines the value of another attribute

Candidate Key:

- A unique identifier. One of the candidate keys will become the primary key
  - e.g., perhaps there is both credit card number and SS# in a table…in this case both are candidate keys.
- Each non-key field is functionally dependent on every candidate key.

First Normal Form

- No multivalued attributes
- Every attribute value is atomic
- The example on the next slide *is not* in 1st Normal Form (multivalued attributes) ➔ it is not a relation.
- The example on the subsequent slide *is* in 1st Normal form.
- ***All relations*** **are in 1st Normal Form.**

Table with multivalued attributes, not in 1st normal form

INVOICE data (Pine Valley Furniture Company)

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2015	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Note: This is NOT a relation.

Table with no multivalued attributes and unique rows, in 1st normal form

INVOICE relation (1NF) (Pine Valley Furniture Company)

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2015	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2015	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2015	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: This is a relation, but not a well-structured one.

Anomalies in this Table

- **Insertion** – if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- **Deletion** – if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price
- **Update** – changing the price of product ID 4 requires update in multiple records

Why do these anomalies exist?

Because there are multiple themes (entity types) in one relation. This results in duplication and an unnecessary dependency between the entities.

SalesStaff						
EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

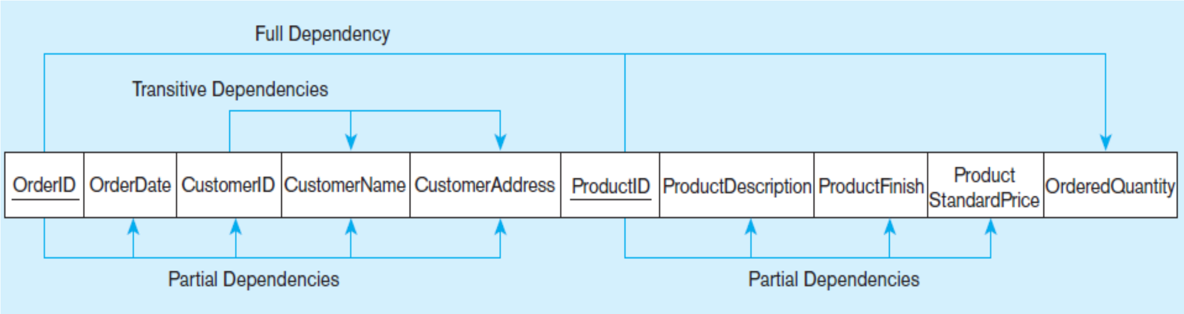
What are the Major Problems with this Approach?

Alternate Approach

Handling Multi-Value Attributes by Adding Fields to the Table

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



OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

CustomerID ➔ CustomerName, CustomerAddress

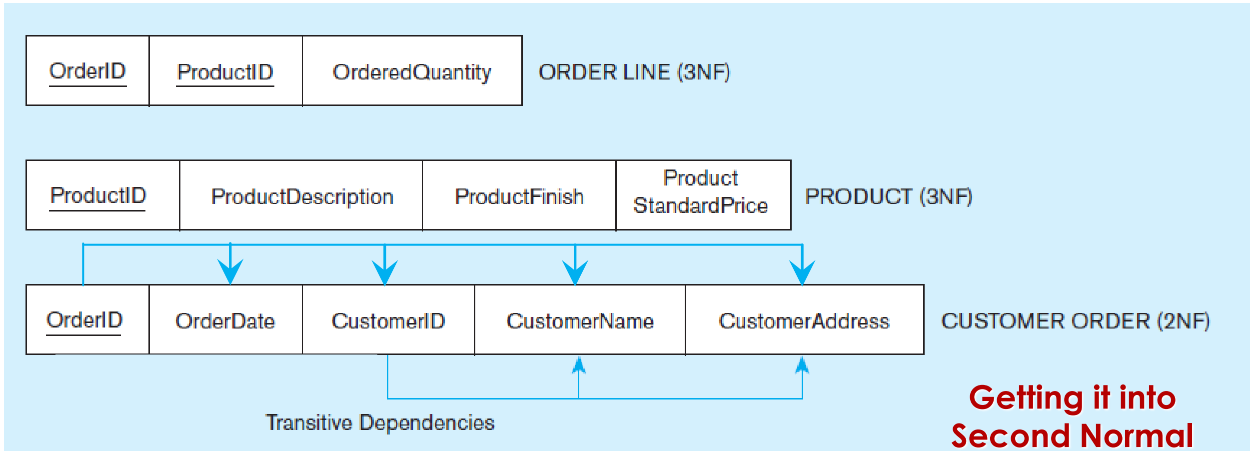
ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

OrderID, ProductID ➔ OrderQuantity

Therefore, NOT in 2nd Normal Form

Functional Dependency Diagram for INVOICE

Removing Partial Dependencies



Getting it into Second Normal Form

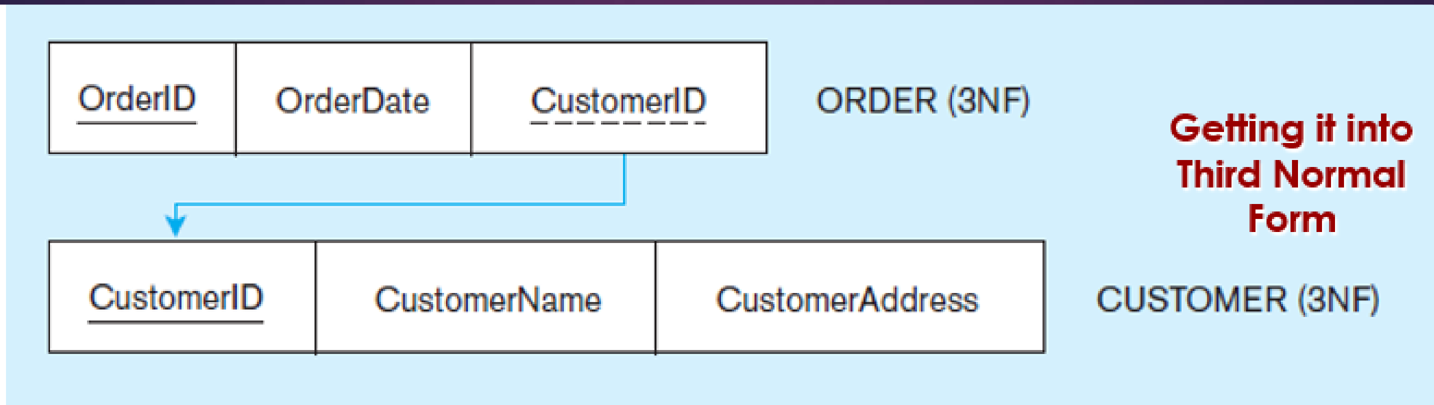
Partial dependencies are removed, but there are still transitive dependencies

Third Normal Form

2NF PLUS ***no transitive dependencies*** (functional dependencies on non-primary-key attributes)

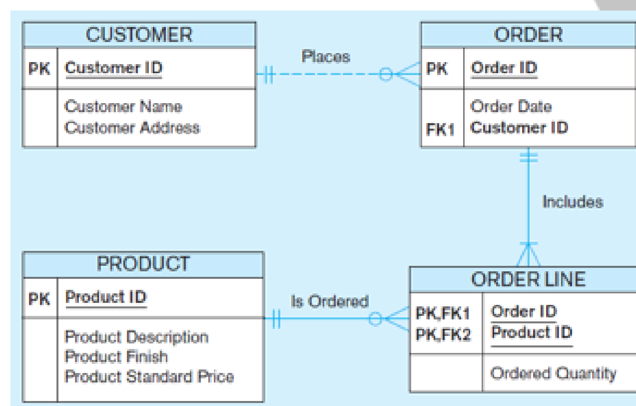
Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third

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 foreign key in the old table



Transitive dependencies are removed.

The following diagram shows the result of normalization, yielding four separate relations where initially there was only one.



Removing Transitive Dependencies

Normalization example

Our first task is to present the data in a tabular format as shown on the next slide.

Looking at this data, we can see that we are not in first normal form because we have no keys, repeating groups and multi-valued fields.

Client	Address	Phone	Pet 1	Pet 2	Pet 3	Pet 4	Visits
Mary Jones	55 Rhodes St	555-290-3083	Boomer, Chihuahua	Trixie, Schnauzer	Fred, Mixed		March 10 at 2:00pm (check up) Boomer, Trixie, Fred March 25 at 8:00am (spay) Trixie
Jerome Franklin	37583 Respite Pines Lane	555-450-4999	Esmerelda, Bulldog				May 27 at 1:00pm (check up) June 15 at 8:00am (grooming) August 5 at 8:00am (grooming)

Normalization example

Normalization example

Things you should consider to understand why this data is not normalized:

What happens when a customer has a fifth pet? Do we re-size the entire database to add that column? What about a sixth, seventh or more?	When most customers only have one or two pets, we still have additional space being used for pet 3, pet 4 and so on.	How do we search for values in a multi-valued field like visits? This can be a processing nightmare and involves a lot of overhead.

Normalization example

How do we get our vet database to first normal form (1NF)? To be in first normal form we need:

- Unique primary key
- One set of values per column
- One value per cell

Normalization example

To improve upon this, we will start by normalizing the data into first normal form.

1NF:

- Each table cell should contain a single value.
- Each record needs to be unique.

Client	Address	Phone	Pet	Breed	Visits
Mary Jones	55 Rhodes St	555-290-3083	Boomer	Chihuahua	March 10 at 2:00pm (check up)
Mary Jones	55 Rhodes St	555-290-3083	Trixie	Schnauzer	March 10 at 2:00pm (check up)
Mary Jones	55 Rhodes St	555-290-3083	Trixie	Schnauzer	March 25 at 8:00am (spay)
Mary Jones	55 Rhodes St	555-290-3083	Fred	Mixed	March 10 at 2:00pm (check up)
Jerome Franklin	37583 Respite Pines Lane	555-450-4999	Esmerelda	Bulldog	May 27 at 1:00pm (check up)
Jerome Franklin	37583 Respite Pines Lane	555-450-4999	Esmerelda	Bulldog	June 15 at 8:00am (grooming)
Jerome Franklin	37583 Respite Pines Lane	555-450-4999	Esmerelda	Bulldog	August 5 at 8:00am (grooming)
Pat Cooper	1250 50th Avenue	555-408-3803	Snots	Mixed	September 12 at 4:00pm (check up)
Pat Cooper	1250 50th Avenue	555-408-3803	Spot	Mixed	
Pat Cooper	1250 50th Avenue	555-408-3803	Sam	Poodle	
Pat Cooper	1250 50th Avenue	555-408-3803	Suzy	Great Dane	September 12 at 4:00pm (check up)

Normalization example

TABLE IN 1NF

Normalization example

When we look at the data normalized to first normal form, we see that we still have some issues.

- Insertion anomalies
  - Data about more than one entity in the relation forces you to insert data about an unrelated entity
- Deletion anomalies
  - Part of the primary key of a row becomes null when the data are deleted, forcing you to remove the entire row. The result of a deletion anomaly is the loss of data that you would like to keep.
- Update anomalies
  - If every row is not changed, then data that should be the same are no longer the same. The potential for these inconsistent data is the modification anomaly

Normalization example

To alleviate some of the issues we find in first normal form, we will continue normalizing the data to second normal form.

2NF:

- The relation is in first normal form.
- All non-key attributes are functionally dependent on the entire primary key.

We start by isolating each group of data into its own entity.

Normalization example

What is functionally dependent upon client id? The information about the client itself (name, address, phone):

Client

ClientID	Client	Address	Phone
1	Mary Jones	55 Rhodes St	555-290-3083
2	Jerome Franklin	37583 Respite Pines Lane	555-450-4999
3	Pat Cooper	1250 50th Avenue	555-408-3803

Normalization example

The pets can be isolated to their own entity as well. We'll use the primary key from the client entity, ClientID, to tie the clients to their pets. Remember – they could have multiple pets. This structure allows any given client to have any number of entries in the pet entity without worrying about having to resize the database again and again.

Pet

PetID	Pet	Breed	ClientID
1	Boomer	Chihuahua	1
2	Trixie	Schnauzer	1
3	Fred	Mixed	1
4	Esmerelda	Bulldog	2
5	Snots	Mixed	3
6	Spot	Mixed	3
7	Sam	Poodle	3
8	Suzy	Great Dane	3

Normalization example

The visits can be isolated to their own entity. We'll use the primary key from the client entity, ClientID, to tie the clients to their visits. Remember – they could have multiple pets. This structure allows any given client to have any number of visit records.

Visit

VisitID	VisitDate	VisitTime	VisitReason	ClientID
1	March 10	2:00pm	Check Up	1
2	March 25	8:00am	Spay	1
3	May 27	1:00pm	Check Up	2
4	June 15	8:00am	Grooming	2
5	August 5	8:00am	Grooming	2
6	September 12	4:00pm	Check Up	3

Normalization example

Next we'll deal with how pets are tied to their visits. Recall that our conceptual diagram depicted a many-to-many relationship between pets and visits. In order to create this type of relationship, we need another table to serve as the go between so that one pet can tie to zero or more visit records and one visit record can tie to one or more pets. We can accomplish this by creating a new table as shown to the right.

PetVisit

PetID	VisitID
1	1
2	1
2	2
3	1
4	3
4	4
4	5
5	6
8	6

Normalization example

- In order to reach third normal form, we are going to break out the pets and their breeds.
- In theory, the vet could store information about various breeds unrelated to the actual client's pets.
- So, we will create a new entity to store breeds and modify the pet entity to relate to it.

Normalization example

3NF

- The relation is in second normal form.
- There are no transitive dependencies.

Normalization example

Pet

PetID	Pet	BreedID	ClientID
1	Boomer	2	1
2	Trixie	6	1
3	Fred	4	1
4	Esmerelda	1	2
5	Snots	4	3
6	Spot	4	3
7	Sam	5	3
8	Suzy	3	3

Breeds

BreedID	Breed
1	Bulldog
2	Chihuahua
3	Great Dane
4	Mixed
5	Poodle
6	Schnauzer
7	Siberian Husky
8	Shih Tzu

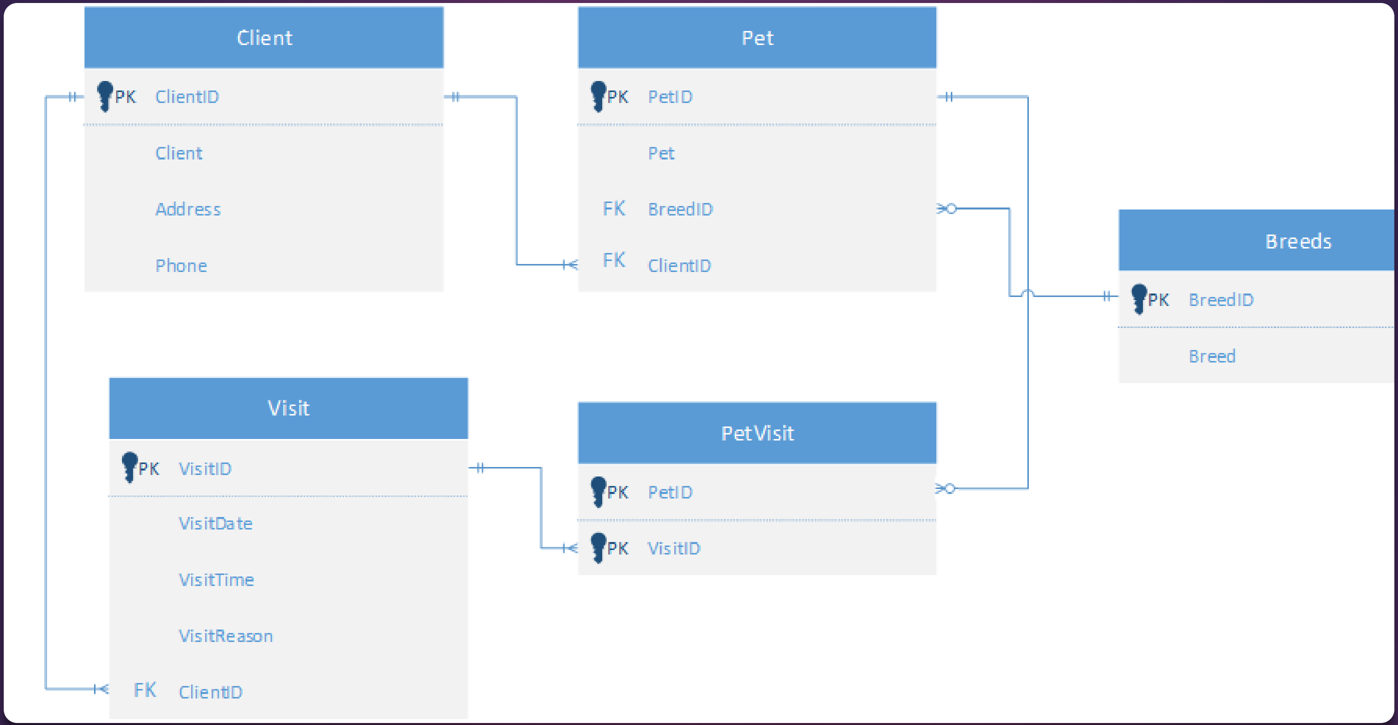
Normalization example

Keep in mind that this part of the modeling process requires you to think about the scenarios that might exist using sample data but you're not dealing with the exact data when designing a database.

Multiply this scenario by thousands of records for any given entity and you'll easily see why we use such small samples to work through this process.

Next, we will document our logical design.

The logical design will depict the entities and relationships but will also include the attributes and primary/foreign key information for this normalized design.



Normalization example