DATABASE DESIGN

THE LOGICAL MODEL AND DATABASE NORMALIZATION

OVERVIEW OF THE FIVE DATABASE SESSIONS

- Session 1: The Transactional Relational Database
  - Work product: Conceptual Model
- **Session 2: Normalizing the Transactional Relational Database**
  - **Work product: Logical Model**
- Session 3: Defining Data Structures Specific to a Database Platform (MariaDB)
  - Work product: Physical Model
- Session 4: Database Initialization Scripts to Create Database & Objects
  - Work Product: SQL scripts to create database objects
- Session 5: SQL Essentials to Query Databases
  - Work Product: SQL commands to query the database

IMPORTANT ICONS

- Database Community Disagreement

- Database Designer Programmer Disagreement

SESSION 2 OBJECTIVES

- What is database normalization?
- What are the first, second, and third normal forms?
- What is a primary key?
- What is a foreign key?

Exercise: Create a logical model from a conceptual model.

DATABASE NORMALIZATION

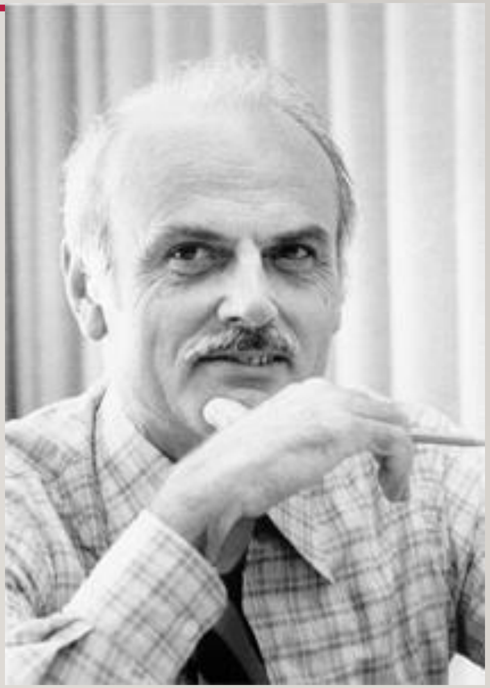
- Database Normalization is the application of mathematical principles to arrange the data in a manner as to prevent insert, update, and delete anomalies as well as to ensure the data can be efficiently queried.

  Note well: it is *not* a means of "saving space." However, as a practical matter each fact in a database will appear in exactly one place (or as we will see with foreign keys, is controlled from only one place).

- In real life all too many databases do not have the necessary rigor applied during the design phase. Inconsistent, incorrect, and missing data is almost always the result.

E.F. “TED” CODD—FATHER OF THE RELATIONAL DATABASE

- Developed the foundational principles while working for IBM.
- Published papers in the early 1970s that form the theoretical foundation of the relational database.



BEFORE FIRST NORMAL FORM (0NF)

- Group all attributes that are related into a single entity.
- Why? Because many examples of the normal forms above third are unrealistic if this is done, which it would have been done in real life.

FIRST NORMAL FORM (1NF)

- A table is in first normal form if it meets four conditions
- First three:
  - 1. Unique Column Names. This is always automatically enforced anyway.
    - Avoid all but the most clear abbreviations. Brevity is good, but clarity is better.
  - 2. Data is atomic.
  - 3. No repeating groups.

1NF EXAMPLE

Name	Address	Birthdate	Home Phone	Cell Phone	Office Phone
Homer Simpson	723 Evergreen Terrace Springfield, OH 42125	May 25, 1970	555-456-7896		
Herman Munster	1313 Mockingbird Lane Transylvania, PA 25968	July 13, 1965	555-896-6548		555-459-2573

- What's wrong?

ATOMIC

First Name	Last Name	Address 1	Address 2	City	State	Zip
Homer	Simpson	723 Evergreen Terrace		Springfield	OH	
Herman	Munster	1313 Mockingbird Lane				

- Is this data atomic?

ANSWER

- It depends.
  - Some databases may benefit from separating the house number from the street. But for most databases this is okay.

- Why is this important?
  - Atomic data is more easily inserted, updated, and queried. In the original example, querying for the city would requiring a full search without benefit of an index. (WHERE address LIKE '%Springfield, OH%')
  - The atomic data is more easily indexed and accurately queried. (WHERE city = 'Springfield' AND state = 'OH')

JSON AND XML DATA TYPES

- Storing JSON and XML data in a table is anything but atomic.
- Programmers like using these data types as they allow them to simply retrieve the value and place it in a web interface.
- Data stored in this manner is far more difficult to query.
- **Bottom line: if this data must be queried, do not use this data type.**

REPEATING GROUPS

- The phone numbers is a repeating group: the same type of data repeating in multiple columns.
- Why is this bad?
  - A search for a phone number would have to be made across multiple columns.
  - The addition of a fourth phone would require alterations to the table and any queries that search for a phone number. No single index will assist with making the query more efficient.

REPEATING GROUPS--SOLUTION

- The phone numbers should be in a separate table.
  - Only one column needs to be searched.
  - There are no holes (NULL or empty string values) in the data for phone numbers that don’t exist.
  - The requirement to add more phones per parent record is simply a row addition requiring no change to the database or the search queries.

Name	Phone Number	Phone Type
Homer Simpson	555-456-7896	Home
Herman Munster	555-896-6548	Home
Herman Munster	555-459-2573	Office

Programmers often dislike this solution as it requires them to make more table joins. DBAs simply have to stand their ground on this.

REPEATING GROUP EXCEPTIONS

- Are there any exceptions?
  - Address1 and Address2
  - Also note: Address2 is often incorrectly stored as a Null value instead on an empty string. We will talk more about this in the third session.

1NF—THE FINAL REQUIREMENT—PRIMARY KEY

- All tables must be given a *primary key*.
- A primary key is that one column (simple) or combination of columns (compound or composite) that will uniquely identify a single record from all the other records in the same table.
- Several types:
  - **Business** or **natural key**. OH standing for Ohio is a natural key. An item or SKU number is another good example of a business key.
  - **Surrogate key**. This is an automatically generated number or unique identifier. This value is usually invisible to the business user.
  - **Business Surrogate key**. This is some kind of automatically generated number that is known and used by the business users One example of a business surrogate key is 1003498 to identify an invoice.

SOURCES OF SURROGATE PRIMARY KEYS

- Manually generated.
  - Useful for small sets.
- Table.
  - Most common source. A column is added to the table that increments by 1 or other predetermined increment.
- Sequence object.
  - Many database platforms allow the creation of a sequence object.
  - The sequence object is used when the key needs to be unique across multiple tables or the entire database.
- Global Unique Identifier (GUID or UUID).
  - This generates a 36 character value that should never repeat.
  - This should be only used for databases that have to be combined with others. In reality it is overused.

PRIMARY KEY WARNING

- Do not use sensitive data as the primary key, such as Social Security Number.
  - Makes it difficult to protect it properly.

PRIMARY KEY GUIDANCE

- Use the business key if a good one is available.
- If one isn't available, use a surrogate key.
- If there are a lot of columns to the key, a surrogate key may be better. Designer call.

SURROGATE KEY EXTREME EXAMPLE

State ID	State Code	State
1	AL	Alabama
2	AK	Alaska
…..	……	…..
35	OH	Ohio
36	OK	Oklahoma
…..	…..	…..
50	WY	Wyoming

- Some DBAs will use this as joining integer columns is faster than joining string columns.
- Using the codes in this case will often remove the need for the join at all.

FIRST NORMAL FORM (1NF)

- A table meeting these four conditions:
  1. Unique Column Names.
  2. Data is atomic.
  3. No repeating groups.
  4. Has a primary key.

  is in First Normal Form.

SECOND NORMAL FORM (2NF)

- A table is in Second Normal Form if:
  - It is in First Normal Form
  - And each non-key attribute depends on the entire primary key.

Order Number	SKU Code	SKU Description	Quantity	Cost per Unit
10058	**6045**	Install Washer	1	150.00
10058	**6088**	Install Dryer	1	175.00
10059	**6045**	Install Washer	1	150.00
10059	**6088**	Install Dryer	1	175.00

In the example above the primary key is the combination of order number and SKU code columns. What's wrong?

  - This is already accomplished with simple keys.

THIRD NORMAL FORM (3NF)

- A table is in Third Normal Form (3NF) if:
  - It is in Second Normal Form,
  - and no non-key attribute has a transitive dependence on any other non-key attribute.

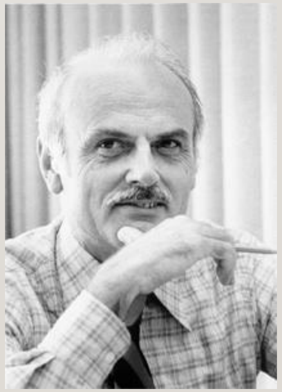
Zip Code	City	State Code	State
98765	Anchorage	AK	Alaska
98764	Anchorage	AK	Alaska
97856	Nome	AK	Alaska
97855	Nome	AK	Alaska

In the example above the zip code column is the primary key. What's wrong?

NORMALIZATION SUMMARY

- The three normal forms can be summarized as follows:
- Each and every non-key attribute depends on
  - the key;   (1NF)
  - the whole key; (2NF)
  - and nothing but the key; (3NF)

  - So help me (E.F.) Codd.



HIGHER LEVEL NORMAL FORMS

- The first three normal forms are core concepts and apply to any transactional database. Almost every database is properly structured when normalized through third normal form.
- There are some very uncommon circumstances beyond the scope of this class that require additional handling and are addressed by higher normal forms. A database designer must be able to recognize these exceptions.

THE BIG TAKEAWAYS

- **Future Programmers:**
  - You don’t need to understand these rules. Just know that there is a reason for the rules.
  - Database Designers are working to ensure that the data can be stored and queried efficiently. They are not simply trying to make your lives more difficult.
  - There is a reason for the rules, and if they aren’t followed, bad things usually happen.
- **Future Database Designers:**
  - You do need to understand how these rules work and why they are important.
  - You have to be confident in your ability to determine and explain why designs employing these principles are best.
  - Sometimes programmers are going to think that you are making their life more difficult for no good reason. You have to be confident in holding your ground.

EXCEPTIONS

- Audit tables are exempt from normalization rules.
    - They are one-time insert only records and do not need to otherwise support normal database transactions.
- ***These rules do not apply to analytical relational databases.***
    - Why bring this up? Because too often data warehouse designers try to apply these rules.

DESIGN ERRORS/ISSUES IN WORK ORDER PRO DATABASE

- Issue #1: How many customers have we had?
- Issue #2: Why isn't the work permits count by state correct?
- Issue #3: Importance of correct column names.

LOGICAL MODEL

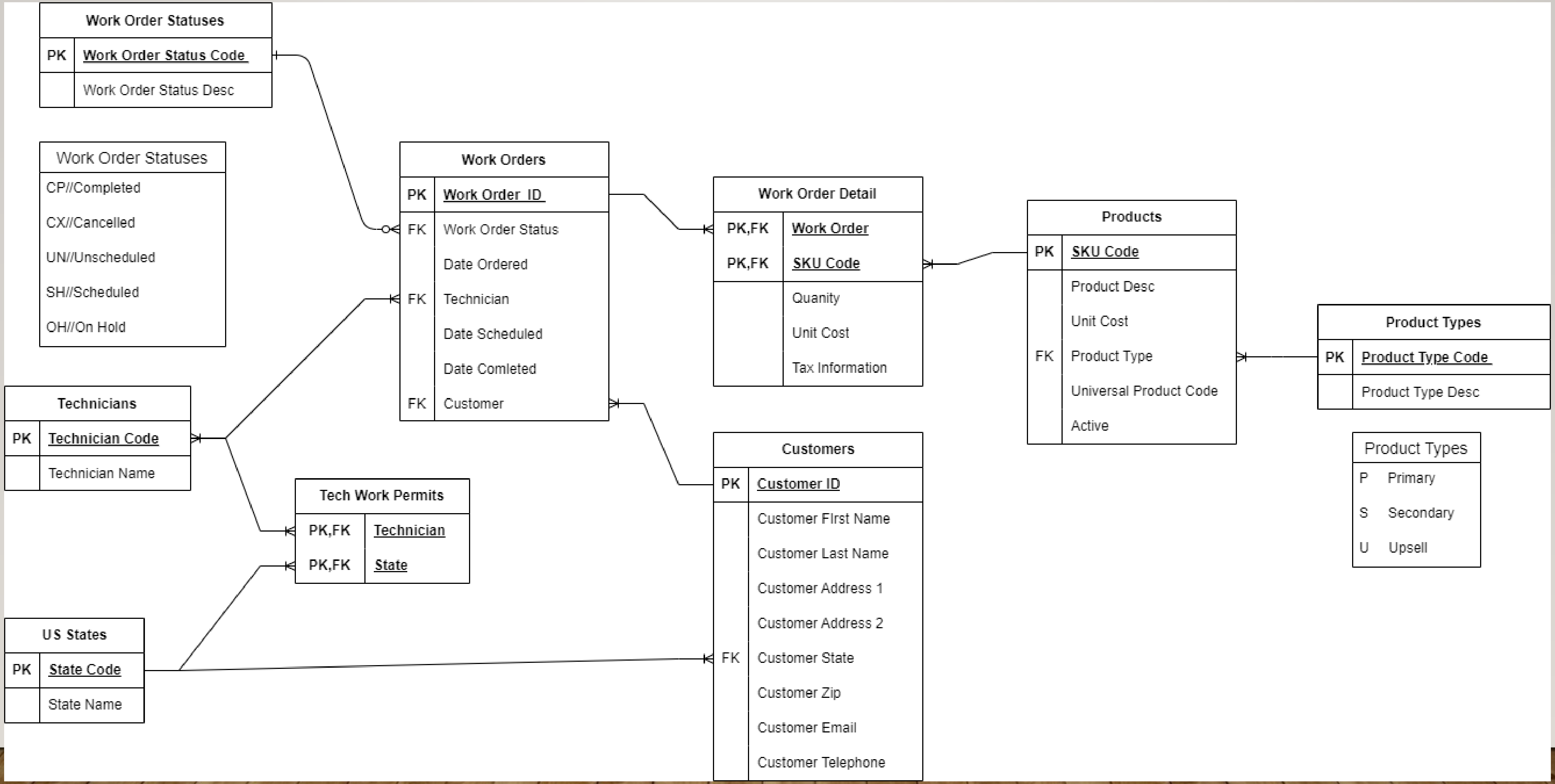
- The Logical Model shows:
  - the primary key and non-key attributes of each entity and relationships between the entities.
  - the relationship between the entities using foreign keys.
- It is platform agnostic.
- It does not need to be understandable by business users.

	Conceptual	**Logical**	**Physical**
Understandable By Business Users	Yes	**No**	No
Database Platform Agnostic	Yes	**Yes**	No

FOREIGN KEYS

- A Foreign Key is column or columns that is a primary key in another table and have been added to establish a parent-child relationship.
- The most common relationship is many-to-one.
  - The table with the foreign key is the many side.
- Entities with a many-to-many relationship cannot be directly connection.
  - Requires at least one intermediary table that have the primary keys from both parent tables.

ADD SURVEY COMPONENT



Work Order Statuses	
PK	Work Order Status Code
	Work Order Status Desc

Work Order Statuses
CP//Completed
CX//Cancelled
UN//Unscheduled
SH//Scheduled
OH//On Hold

Work Orders	
PK	Work Order ID
FK	Work Order Status
	Date Ordered
FK	Technician
	Date Scheduled
	Date Comleted
FK	Customer

Work Order Detail	
PK,FK	Work Order
PK,FK	SKU Code
	Quanity
	Unit Cost
	Tax Information

Products	
PK	SKU Code
	Product Desc
	Unit Cost
FK	Product Type
	Universal Product Code
	Active

Product Types	
PK	Product Type Code
	Product Type Desc

Product Types	
P	Primary
S	Secondary
U	Upsell

Technicians	
PK	Technician Code
	Technician Name

Tech Work Permits	
PK,FK	Technician
PK,FK	State

US States	
PK	State Code
	State Name

Customers	
PK	Customer ID
	Customer FIrst Name
	Customer Last Name
	Customer Address 1
	Customer Address 2
FK	Customer State
	Customer Zip
	Customer Email
	Customer Telephone

GROUP LOGICAL MODELS

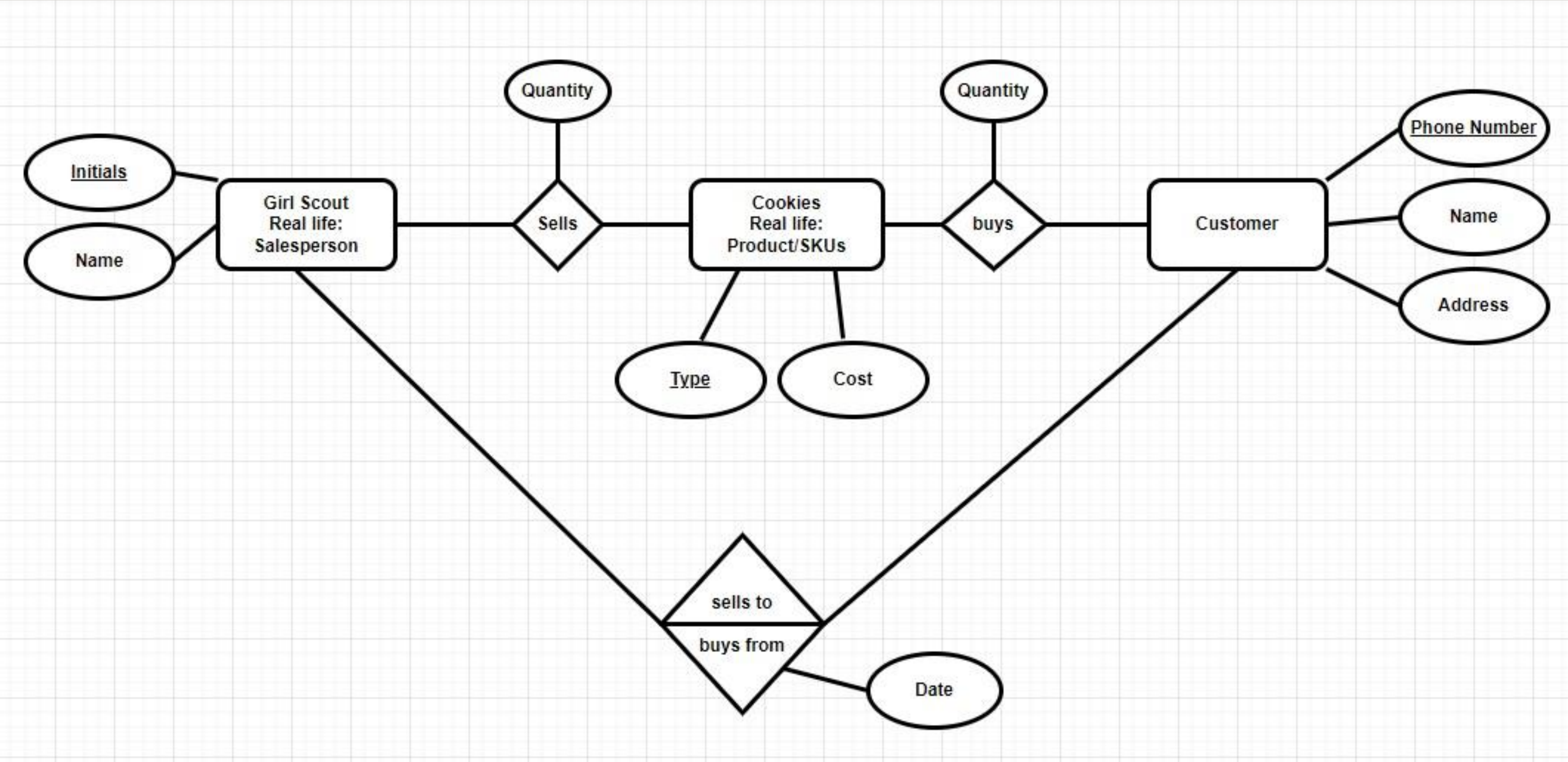
- Work in your groups to create a group logical model from your individual logical models.

DENORMALIZATION

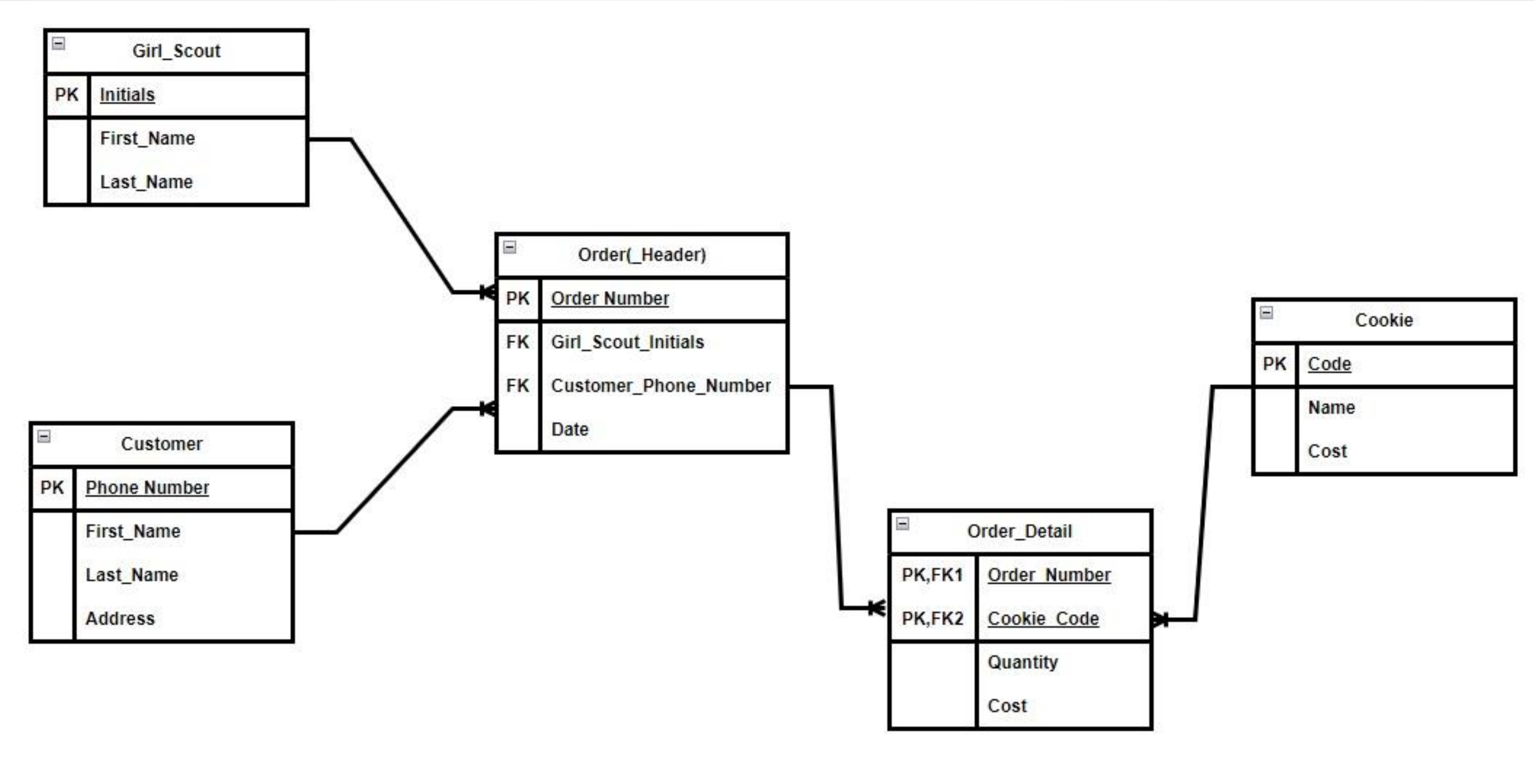
- Denormalization is the intentional introduction of a normalization error.
  - Reasons include performance or easier query writing.
- This should never happen at design time. It is done only to resolve performance issues that cannot be resolved in any other way.

- I have denormalized once or twice in over twenty years.

RECAP – CONCEPTUAL MODEL



RECAP – LOGICAL MODEL



	Girl_Scout
PK	Initials
	First_Name
	Last_Name

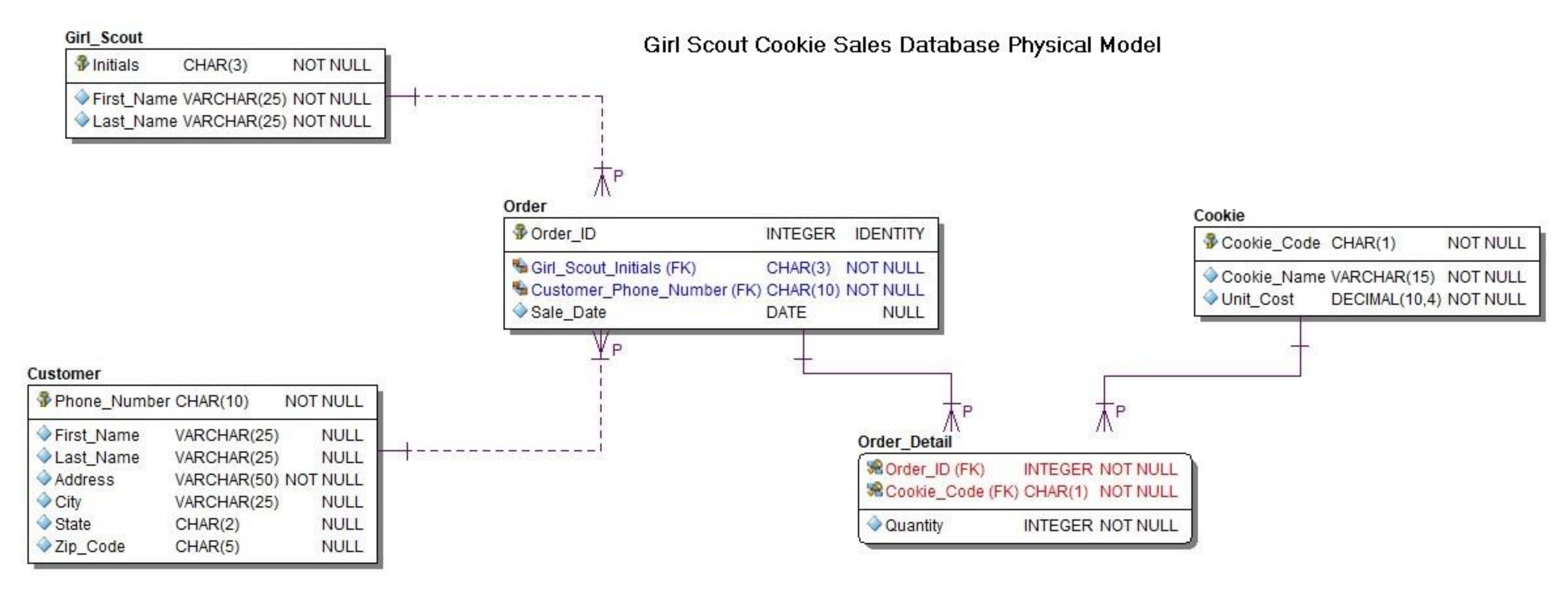
	Order(_Header)
PK	Order Number
FK	Girl_Scout_Initials
FK	Customer_Phone_Number
	Date

	Customer
PK	Phone Number
	First_Name
	Last_Name
	Address

	Cookie
PK	Code
	Name
	Cost

	Order_Detail
PK,FK1	Order_Number
PK,FK2	Cookie_Code
	Quantity
	Cost

HOMEWORK – LOGICAL TO PHYSICAL MODEL



Girl Scout Cookie Sales Database Physical Model

Girl_Scout

Initials	CHAR(3)	NOT NULL
First_Name	VARCHAR(25)	NOT NULL
Last_Name	VARCHAR(25)	NOT NULL

Order

Order_ID	INTEGER	IDENTITY
Girl_Scout_Initials (FK)	CHAR(3)	NOT NULL
Customer_Phone_Number (FK)	CHAR(10)	NOT NULL
Sale_Date	DATE	NULL

Cookie

Cookie_Code	CHAR(1)	NOT NULL
Cookie_Name	VARCHAR(15)	NOT NULL
Unit_Cost	DECIMAL(10,4)	NOT NULL

Customer

Phone_Number	CHAR(10)	NOT NULL
First_Name	VARCHAR(25)	NULL
Last_Name	VARCHAR(25)	NULL
Address	VARCHAR(50)	NOT NULL
City	VARCHAR(25)	NULL
State	CHAR(2)	NULL
Zip_Code	CHAR(5)	NULL

Order_Detail

Order_ID (FK)	INTEGER	NOT NULL
Cookie_Code (FK)	CHAR(1)	NOT NULL
Quantity	INTEGER	NOT NULL