Relational Databases

Design Theory

8 February 2024

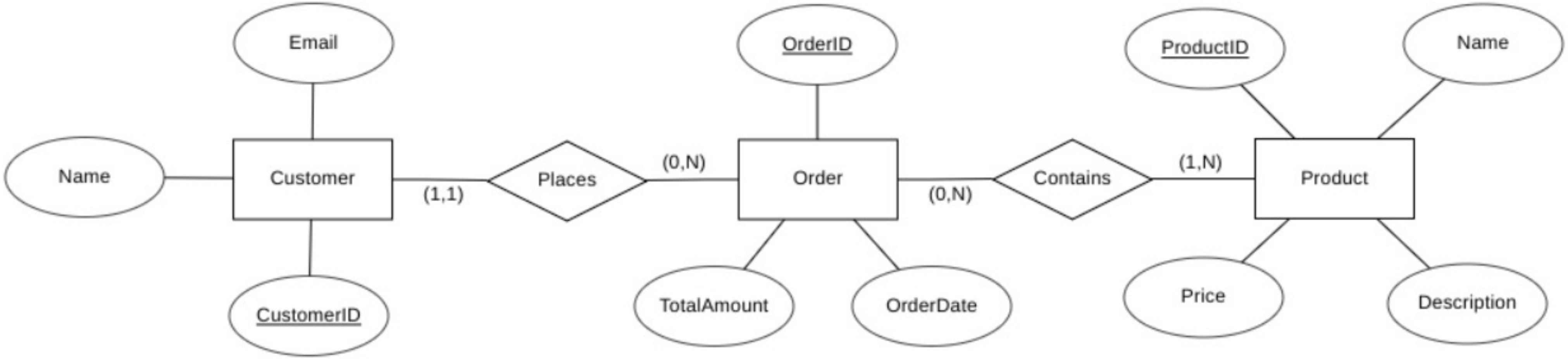
克明峻德，格物致知

Quick Recap: Entity-Relationship Diagram

Key Concepts:

- ERDs are foundational in visualizing and structuring the data relationships in a database, facilitating clear communication and planning in the design process. They highlight entities, their attributes, and the relationships between them, crucial for understanding the database's conceptual layout.
- One-to-One, One-to-Many, Many-to-Many, Self-Referencing, and M-way are critical in ERDs. These relationships help in accurately modeling complex real-world interactions within the database.

ERD Example



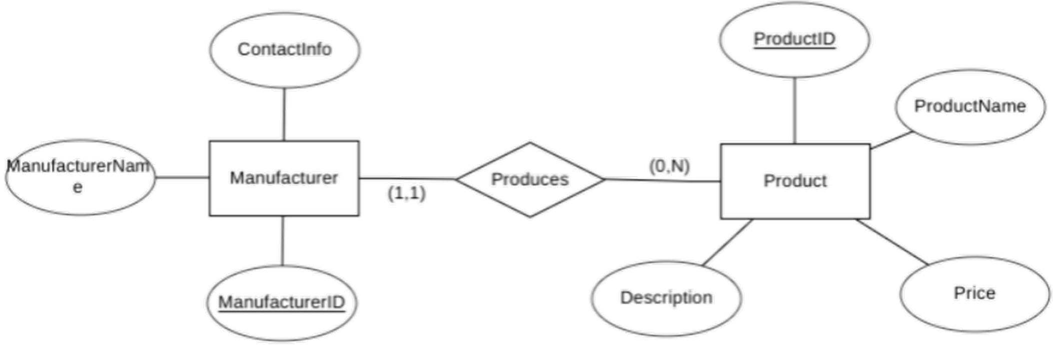
- **Scenario:** Customer's Order History.

Transfroming ER Models to Relational Schemas

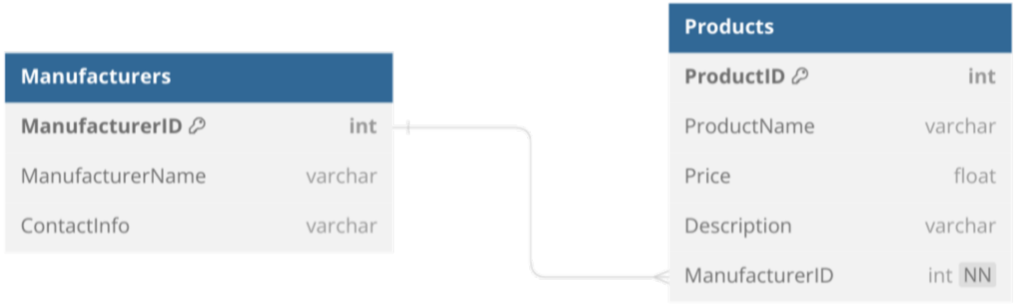
Key Points:

- **Entity Conversion:** Each entity in the ER model becomes a table in the schema.
- **Relationship Handling:** Relationships in ER models are translated into foreign keys and join tables.
- **Attribute Translation:** Attributes of entities in the ER model become the columns of the tables in the schema.

ER Diagram:



Relational Schema:



Manufacturers	
ManufacturerID	int
ManufacturerName	varchar
ContactInfo	varchar

Products	
ProductID	int
ProductName	varchar
Price	float
Description	varchar
ManufacturerID	int NN

Best Practices:

- **Consistency:** Keep names and types consistent during the transition.
- **Integrity Constraints:** Enforce data integrity through constraints derived from the ER model (e.g., not null, unique).

Normalisation Consideration

- The schema might needs normalisation to ensure efficiency and data integrity.

What Makes a Good Data Model?

Efficient Representation of Data Relationships

- A good data model accurately represents the relationships between different data entities, ensuring clarity and logical structure.

Minimised Redundancy

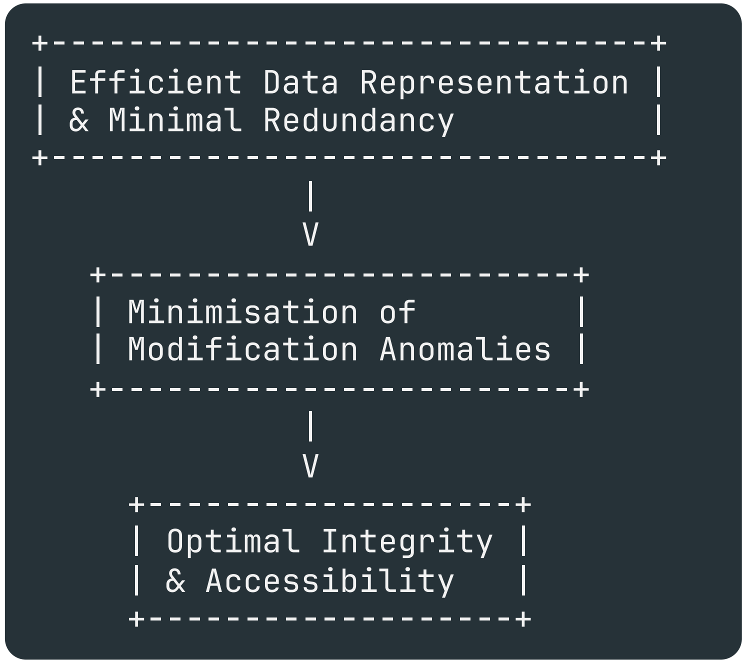
- Reduces unnecessary repetition of data across the database, enhancing efficiency and storage utilization.

Minimised Modification Anomalies

- Effectively manages and minimizes potential errors in data modification, such as update, insertion, and deletion anomalies.

Data Integrity and Accessibility

- Maintains high data integrity and provides ease of access and use, making the database reliable and user-friendly.



```
+-------------------------------+
| Efficient Data Representation |
| & Minimal Redundancy          |
+-------------------------------+
              |
              V
   +------------------------+
   | Minimisation of        |
   | Modification Anomalies |
   +------------------------+
              |
              V
     +-------------------+
     | Optimal Integrity |
     | & Accessibility   |
     +-------------------+
```

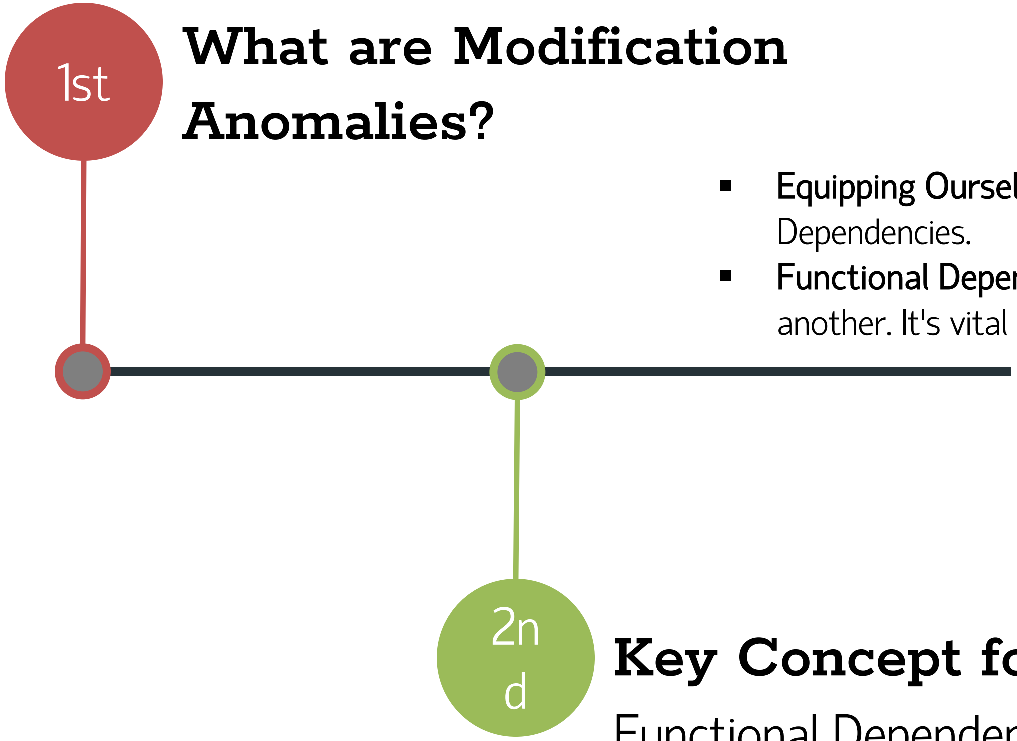
Learning Path

1st

What are Modification Anomalies?

- **The Foundation:** Before managing modification anomalies, it's essential to know what they are. These are issues in databases that lead to inconsistent data during Insertions, Deletions, and Updates.
- **Why It Matters:** Identifying these anomalies is the first critical step. It's about understanding the challenges in database operations to develop effective strategies for handling them.

Learning Path



1st

What are Modification Anomalies?

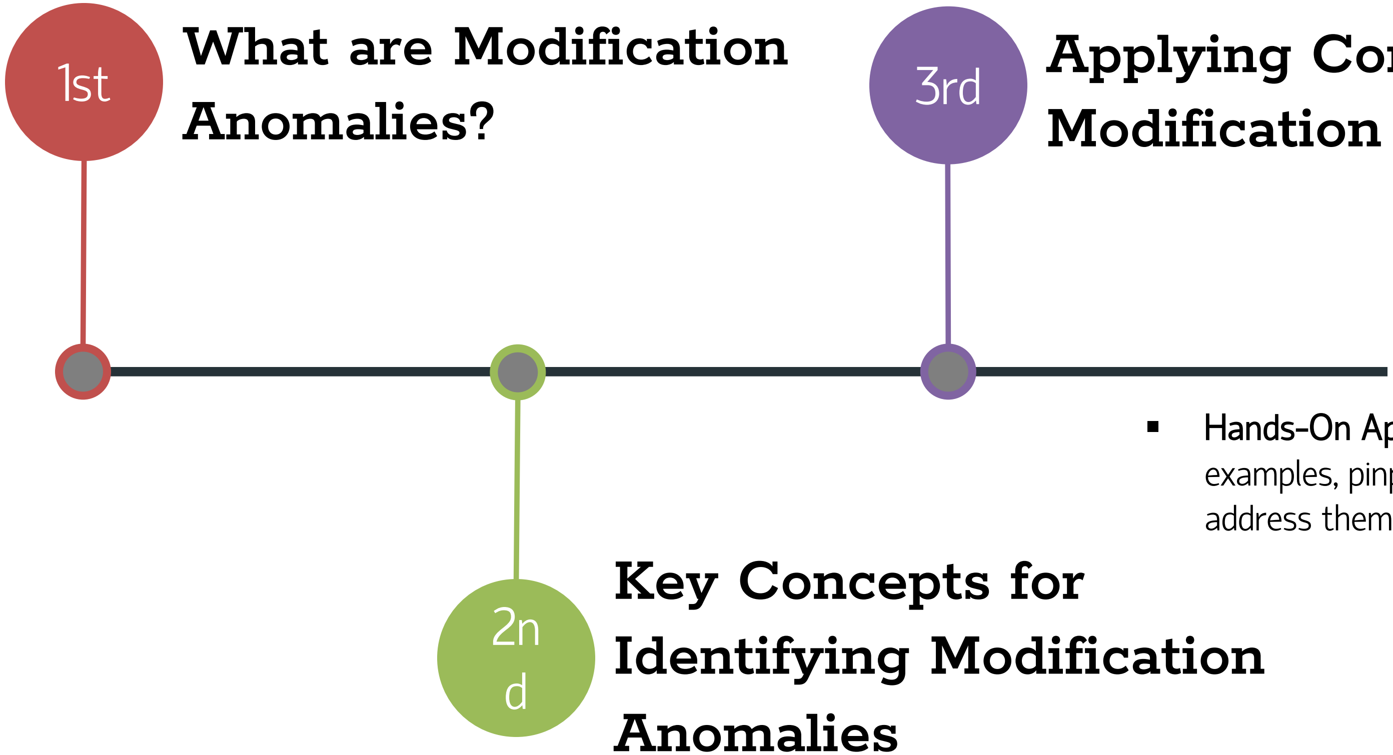
- **Equipping Ourselves:** To handle modification anomalies, we equip ourselves with a key concept: Functional Dependencies.
- **Functional Dependencies:** This concept helps us understand how one set of data in our database affects another. It's vital for identifying potential spots where anomalies might occur.

2n
d

Key Concept for Identifying Modification Anomalies

Functional Dependencies

Learning Path



1st **What are Modification Anomalies?**

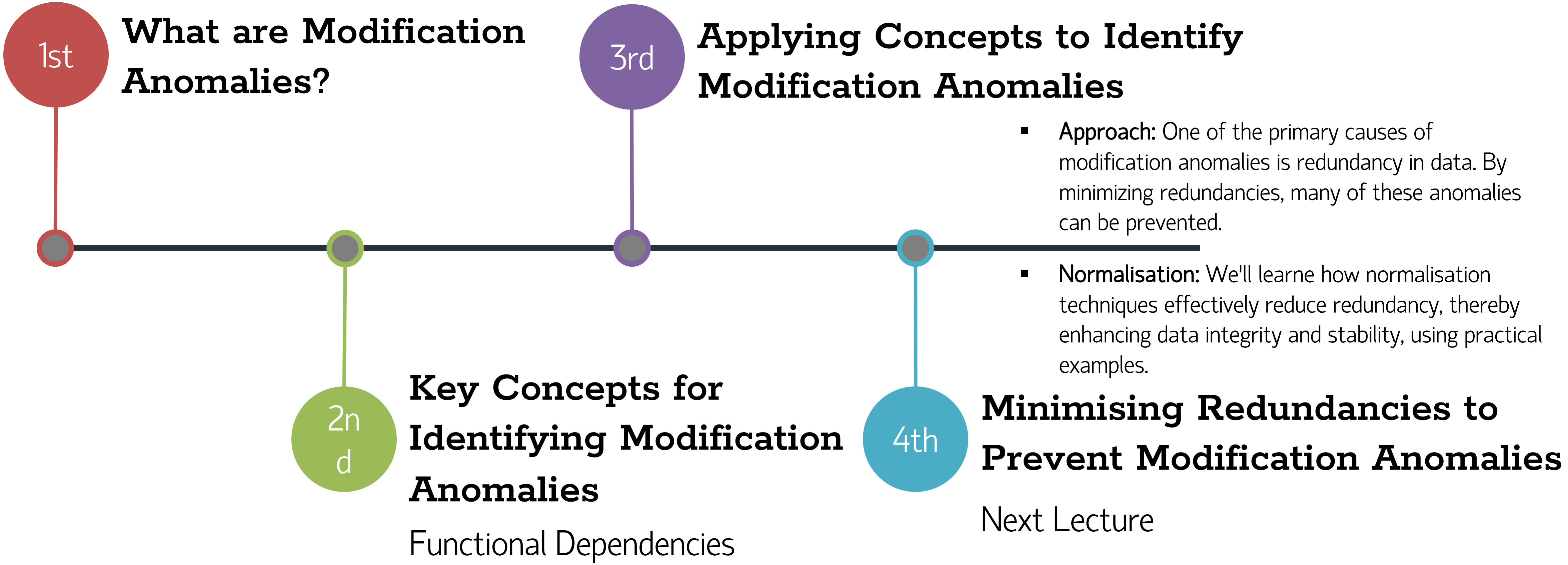
2n
d **Key Concepts for Identifying Modification Anomalies**

Functional Dependencies

3rd **Applying Concepts to Identify Modification Anomalies**

- **Hands-On Application:** We'll use Functional Dependencies to analyse real-life examples, pinpointing where anomalies are likely to arise and exploring ways to address them.

Learning Path



1st

What are Modification Anomalies?

2nd

Key Concepts for Identifying Modification Anomalies

Functional Dependencies

3rd

Applying Concepts to Identify Modification Anomalies

- **Approach:** One of the primary causes of modification anomalies is redundancy in data. By minimizing redundancies, many of these anomalies can be prevented.
- **Normalisation:** We'll learne how normalisation techniques effectively reduce redundancy, thereby enhancing data integrity and stability, using practical examples.

4th

Minimising Redundancies to Prevent Modification Anomalies

Next Lecture

Learning Path

1st

What are Modification Anomalies?

- **The Foundation:** Before managing modification anomalies, it's essential to know what they are. These are issues in databases that lead to inconsistent data during Insertions, Deletions, and Updates.
- **Why It Matters:** Identifying these anomalies is the first critical step. It's about understanding the challenges in database operations to develop effective strategies for handling them.

Modification Anomalies

Key Points:

- Modificaiton Anomalies are unintended side effects in the database system, often arising from inadequate or flaw design.
- These lead to inconsistent data, redundant or duplicated entries and impact in overall efficiency and performance of the database

Types of Anomalies

- **Insertion Anomaly**, **Deletion Anomaly** and **Update Anomaly**

Insertion Anomalies

Definition

- An insertion anomaly occurs in a database when the addition of new data is hindered or results in incomplete records due to the reliance on other related data being present.

Example Scenario

- **Context:** In an e-commerce database with combined product and supplier details, adding a new supplier without associated products leads to records with NULL values for product columns.

```
| Product ID | Product Name         | Supplier ID | Supplier Name    | Supplier Contact   |
|------------|----------------------|-------------|------------------|--------------------|
| 1001       | Solar String Lights | S01         | EcoLights Co.    | +1234567890        |
| NULL       | NULL                 | S03         | NewEco Inc.      | +1122334455        |
```

Explanation

- In this scenario, adding 'NewEco Inc.' as a new supplier without a corresponding product creates entries with NULL values for product fields. This occurrence, typical in denormalized databases, exemplifies an insertion anomaly: you can add supplier details, but the absence of related product data leads to incomplete records.

Deletion Anomalies

Definition

- A deletion anomaly occurs when removing a record from a database inadvertently leads to the loss of additional, valuable data.

Example Scenario

- **Context:** With product details and their supplier information are stored in the same table. When the last product from a specific supplier is deleted, the supplier's details are also lost from the database

```
| Product ID | Product Name         | Supplier ID | Supplier Name    | Supplier Contact   |
|------------|----------------------|-------------|------------------|--------------------|
| 1001       | Solar String Lights  | S01         | EcoLights Co.    | +1234567890        |
| 1002       | LED Lantern          | S02         | BrightLumen Inc.| +1987654321        |
```

Deletion Action

- Delete 'Solar String Lights' (the only product supplied by 'EcoLights Co.')

```
| Product ID | Product Name         | Supplier ID | Supplier Name    | Supplier Contact   |
|------------|----------------------|-------------|------------------|--------------------|
| 1002       | LED Lantern          | S02         | BrightLumen Inc.| +1987654321        |
```

Explanation

- By deleting 'Solar String Lights', all information about its supplier 'EcoLights Co.' is also removed from the table. This deletion anomaly leads to the loss of important supplier details that might be needed for future reference or orders.

Update Anomalies

Definition

- Update anomalies occur when changes made to data in one part of a denormalized database are not consistently replicated in other related parts, leading to data inconsistencies or redundancies.

Example Scenario

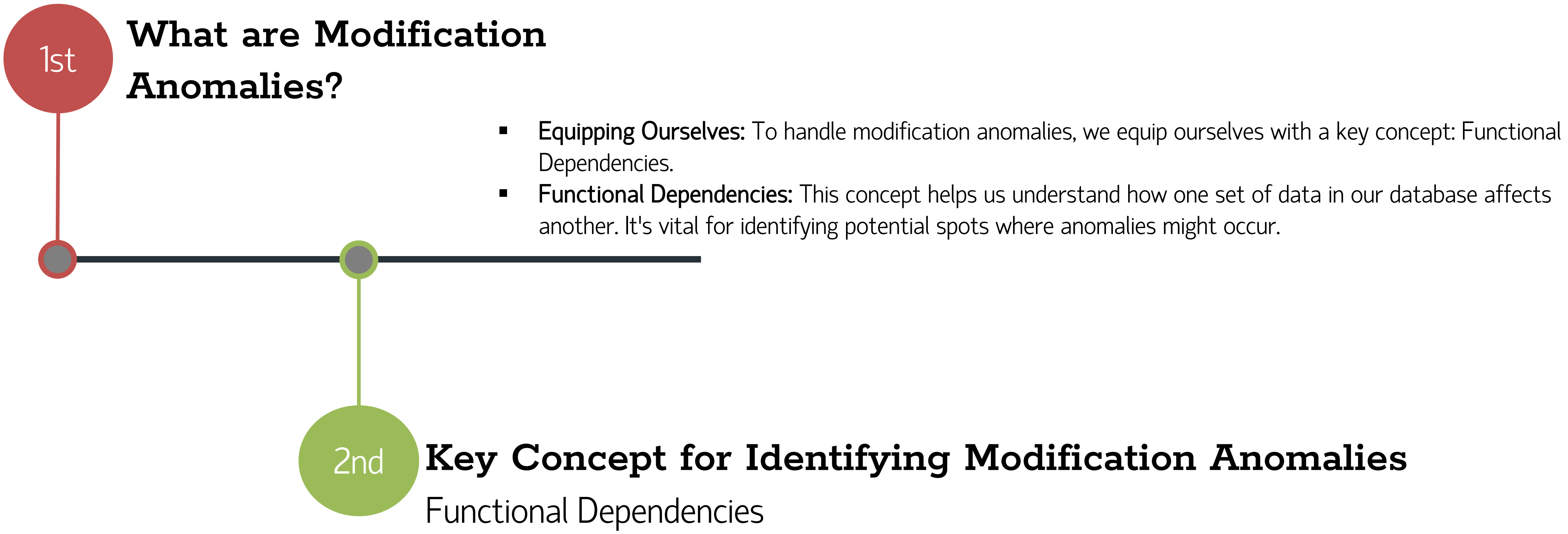
- **Context:** Consider an database where each product's record includes supplier details. An update anomaly happens when changing a supplier's contact information in one product record does not automatically update the same information in other products supplied by the same supplier.

```
| Product ID | Product Name         | Supplier ID | Supplier Name    | Supplier Contact |
|------------|----------------------|-------------|------------------|------------------|
| 1001       | Solar String Lights | S01         | EcoLights Co.    | +1234567890      |
| 1003       | Eco Desk Lamp        | S01         | EcoLights Co.    | Old Contact Info |
```

Explanation

- In this example, updating the contact information for 'EcoLights Co.' in the record for 'Solar String Lights' doesn't update the same information for 'Eco Desk Lamp'. This inconsistency in supplier contact details across different product records is a classic update anomaly.

Learning Path



1st

What are Modification Anomalies?

- **Equipping Ourselves:** To handle modification anomalies, we equip ourselves with a key concept: Functional Dependencies.
- **Functional Dependencies:** This concept helps us understand how one set of data in our database affects another. It's vital for identifying potential spots where anomalies might occur.

2nd

Key Concept for Identifying Modification Anomalies

Functional Dependencies

Functional Dependencies

Definition

- Functional Dependencies are relationships where one set of data in a database uniquely determines another set. These relationships are key to structuring databases effectively.
- **Importance:** Understanding FDs helps in creating logically organized and efficient databases.

Example Scenario

- Customer Table:

```
+------------+-------------+-----------------+
| CustomerID | Name        | Address         |
+------------+-------------+-----------------+
| 1001       | Alice Smith | 123 Apple Lane  |
| 1002       | Bob Jones   | 456 Berry Blvd. |
| 1003       | Carol White | 789 Cherry St.  |
+------------+-------------+-----------------+
```

- List of Dependencies:

```
CustomerID → CustomerName, Address
```

E-Commerce Example

Scenario

- Product Table:

```
| SKU Number | Name                | Price | Description                |
|------------|--------------------|-------|---------------------------|
| SKU1234    | Wireless Mouse      | 29.99 | Ergonomic wireless mouse   |
| SKU5678    | Bluetooth Keyboard | 49.99 | Compact Bluetooth keyboard|
| SKU9101    | USB-C Hub           | 39.99 | 4-port USB-C hub           |
```

- List of Dependencies:

```
SKU Number → Name, Price, Description
```

Explanation

- A single SKU Number uniquely determines a product's Price, Name, Description, and Category.
- This dependency ensures consistent and reliable data across the database, which is essential for inventory management, customer experience, and analytics.

Identifying Functional Dependencies

The Process of Identifying Functional Dependencies

- Identifying FDs often involves looking for patterns and consistent relationships within your data.

- Example Scenario: (Sales Table)

```
| Order ID | Order Date | Customer ID | Total Amount | Product ID |
|----------|------------|-------------|--------------|------------|
| O1001    | 2021-07-01 | C123        | 150.00       | P001       |
| O1002    | 2021-07-02 | C124        | 200.00       | P002       |
| O1003    | 2021-07-02 | C125        | 250.00       | P003       |
```

- Practical Tips:
  - **Unique Identifiers:** Start with primary keys or unique identifiers as they often determine other information.
  - **One-to-One Relationship:** Look for cases where one data element always corresponds to a specific value of another.
  - **Real-World Logic:** Apply your understanding of the real-world context. For instance, in a sales database, an 'Order ID' will determine 'Order Date' and 'Total Amount'."

- List of Dependencies:

```
Order ID → Order Date, Customer ID, Total Amount, Product ID
```

Identifying Functional Dependencies (cont.)

- Example Scenario: (Sales Table)

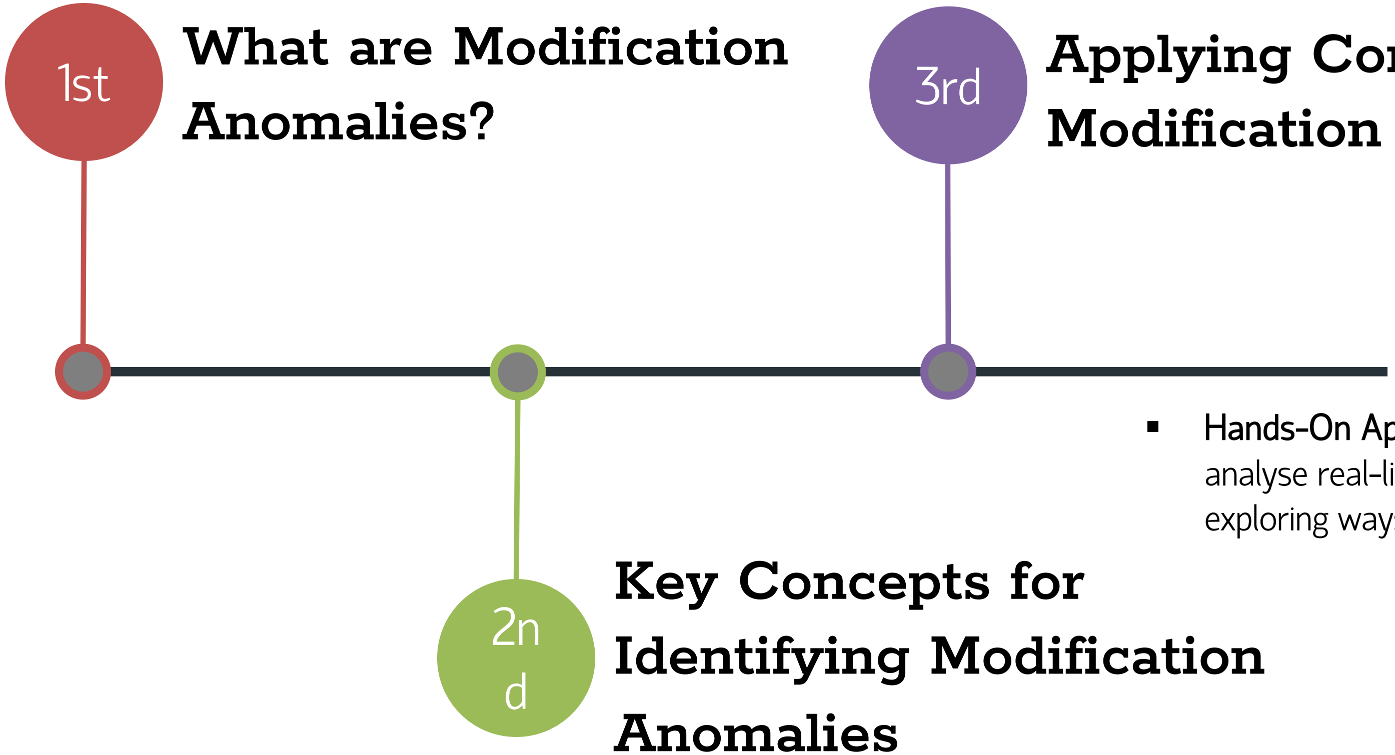
```
| Order ID | Product ID | Product Name         | Customer ID | Customer Name | Order Total |
|----------|------------|----------------------|-------------|---------------|-------------|
| O1001    | P001       | Wireless Mouse       | C123        | Alice         | 300.00      |
| O1001    | P002       | Bluetooth Keyboard   | C123        | Alice         | 300.00      |
| O1002    | P002       | Bluetooth Keyboard   | C124        | Bob           | 200.00      |
| O1003    | P003       | USB-C Hub            | C125        | Carol         | 450.00      |
| O1003    | P001       | Wireless Mouse       | C125        | Carol         | 450.00      |
```

- List of Dependencies:

```
Order ID → Order Total, Customer ID, Customer Name
Product ID → Product Name
Customer ID → Customer Name
```

- The current table's structure, where Order ID and Product ID are in the same table without a clear functional dependency, can lead to data redundancy and complications in maintaining the database, as seen with repeated customer and order total information.
- In a well-structured database, a separate junction table or line items table would typically be used to appropriately represent this many-to-many relationship.

Learning Path



What are Modification Anomalies?

Applying Concepts to Identify Modification Anomalies

- **Hands-On Application:** We'll use Functional Dependencies and Closure to analyse real-life examples, pinpointing where anomalies are likely to arise and exploring ways to address them.

Key Concepts for Identifying Modification Anomalies

Functional Dependencies and Closure

Recap and Beyond: Functional Dependencies

- **Focus:** This is about the relationship between sets of attributes in a database. Specifically, it indicates that if you know the value of one set of attributes, you can determine the value of another set.

- **Modification Anomalies:** By structuring tables based on FDs, update/insert/delete anomalies can be minimized, as each piece of information is stored only once and updated consistently.

Functional Dependencies: Identifying Update Anomalies

Order Table Example

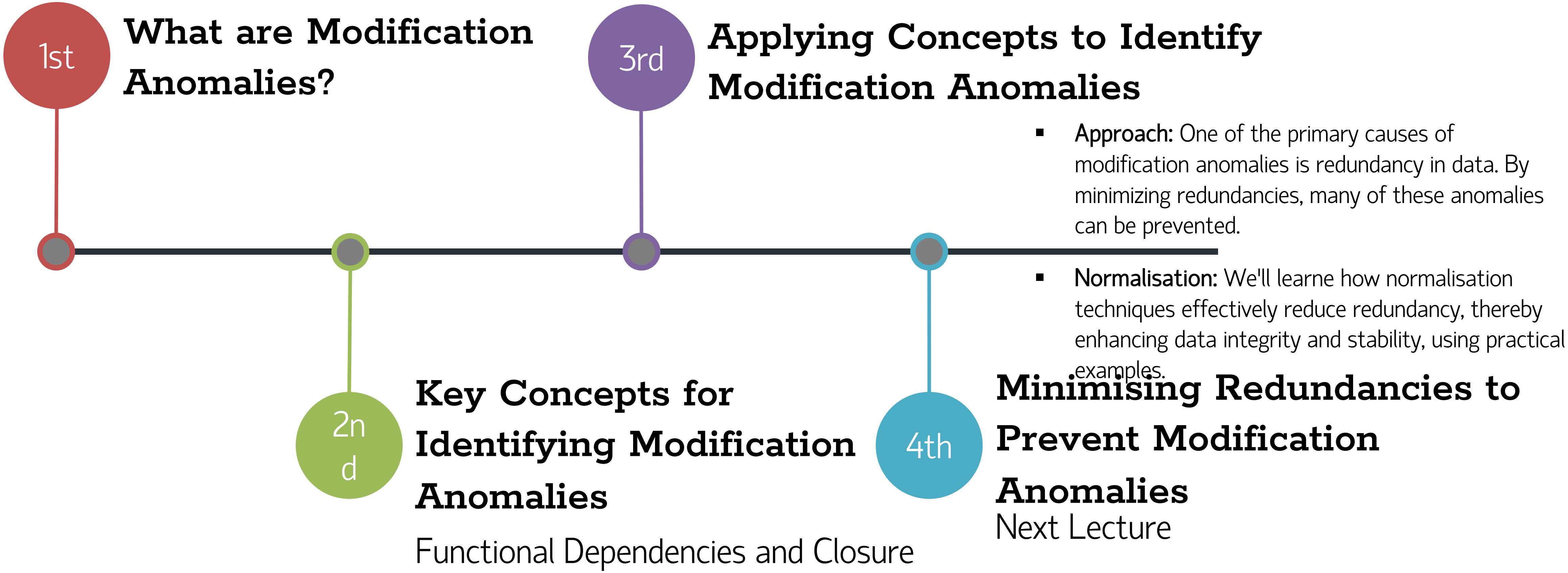
```
| CustomerID | CustomerName | Address          | OrderID | OrderDate  | OrderAmount |
|------------|--------------|------------------|---------|------------|-------------|
| C001       | Alice Smith  | 123 Apple Street | O100    | 2021-07-15 | $150        |
| C001       | Alice Smith  | 123 Apple Street | O101    | 2021-07-20 | $200        |
| C002       | Bob Johnson  | 456 Berry Avenue | O102    | 2021-07-18 | $250        |
```

Functional Dependency and Anomaly Analysis

```
| Attribute Set | Functional Dependency   | Anomaly Detection                                  |
|---------------|-------------------------|----------------------------------------------------|
| CustomerID    | CustomerID →            | Potential Update Anomaly if `CustomerName` or      |
|               | CustomerName, Address   | `Address` varies for the same `CustomerID`         |
| OrderID       | OrderID → OrderDate,    | No apparent anomaly; `OrderID` uniquely determines |
|               | OrderAmount, CustomerID | `OrderDate`, `OrderAmount`, and `CustomerID`       |
```

- **Anomaly Detection Pointer:** In summary, seeing a supposed-to-be unique identifier like CustomerID and OrderID repeating in a dataset is a flag for a potential anomaly that should be investigated further.
- **Scenario:** Imagine a scenario where, in future data entries, 'CustomerName' varies for the same 'CustomerID', which would be an update anomaly. This inconsistency would indicate a violation of the functional dependency ('CustomerID' ⟶ 'CustomerName', 'Address').

Learning Path



1st **What are Modification Anomalies?**

2nd **Key Concepts for Identifying Modification Anomalies**

Functional Dependencies and Closure

3rd **Applying Concepts to Identify Modification Anomalies**

- **Approach:** One of the primary causes of modification anomalies is redundancy in data. By minimizing redundancies, many of these anomalies can be prevented.
- **Normalisation:** We'll learne how normalisation techniques effectively reduce redundancy, thereby enhancing data integrity and stability, using practical examples.

4th **Minimising Redundancies to Prevent Modification Anomalies**

Next Lecture

Recap and Key Takeaways

- A good data model is characterized by its efficiency in representing data relationships and its effectiveness in minimizing modification anomalies, ensuring data consistency and integrity across the database.
- Functional dependencies and closure serve as crucial metrics for identifying potential anomalies. These insights guide database design, helping to structure data in a way that enhances reliability and scalability while minimizing redundancy and data anomalies.

```
| CustomerID | CustomerName    | Address               |
|------------|-----------------|-----------------------|
| C001       | Alice Smith     | 123 Apple Street      |
| C002       | Bob Johnson     | 456 Berry Avenue      |
| C001       | Alice Smith     | 789 Cherry Lane       | ←!— Anomaly ⟶
```

- An anomaly is observed with CustomerID C001. It is associated with two different addresses ('123 Apple Street' and '789 Cherry Lane'), which violates the functional dependency. This inconsistency suggests an update anomaly.

Preparing for Normalisation:

- Next Lecture Preview:
  - Understand key methods for organizing database structures to enhance efficiency and minimise modification anomalies.
  - Explore various normalization forms and their significance in optimizing database design and functionality.