Задание 2. Пример 1. Отношение в 1НФ, но не в 2НФ

Предметная область: Online-store

1. Attributes of the Original Relation

* Product: Product ID, Name, Category, Price, Supplier ID
* Supplier: Supplier ID, Name, Contact Information
* Customer: Customer ID, Name, Contact Information, Address
* Order: Order ID, Customer ID, Order Date, Shipping Date, Total Price
* Order Details: Order ID, Product ID, Quantity, Price per Item

2. Non-Trivial Functional Dependencies

* Product ID → {Name, Category, Price, Supplier ID}
* Supplier ID → {Name, Contact Information}
* Customer ID → {Name, Contact Information, Address}
* Order ID → {Customer ID, Order Date, Shipping Date, Total Price}
* {Order ID, Product ID} → {Quantity, Price per Item}

3. Potential Keys of the Original Relation

* Product: Product ID
* Supplier: Supplier ID
* Customer: Customer ID
* Order: Order ID
* Order Details: {Order ID, Product ID}

4. Determinants of Non-Trivial Functional Dependencies

* Product ID, Supplier ID, Customer ID, Order ID, {Order ID, Product ID}

5. Normalization Forms

* 1NF (First Normal Form): The relation is in 1NF as each attribute contains only atomic values.
* 2NF (Second Normal Form): The relation is not in 2NF as some non-key attributes are not fully functionally dependent on the primary key.
* 3NF (Third Normal Form): The relation is not in 3NF due to transitive dependencies.
* BCNF (Boyce-Codd Normal Form): Not applicable as the relation is not in 3NF.
* 4NF (Fourth Normal Form): Not applicable as the relation is not in BCNF.
* 5NF (Fifth Normal Form): Not applicable as the relation is not in 4NF.

6. Illustration of Anomalies

* Insert Anomaly

An insert anomaly occurs when certain data cannot be inserted into the database due to the absence of other data.

Example: Imagine a situation where a new product is ready to be listed on the online store, but the supplier information is not yet finalized. If the database design requires that every product entry must have a valid supplier ID (as a foreign key), this would prevent the new product from being added to the database until the supplier details are available. This can delay listing new products and impact business operations.

| **Product ID** | **Name** | **Category** | **Price** | **Supplier ID** |
| --- | --- | --- | --- | --- |
| P001 | T-Shirt | Clothing | $20 | S001 |
| P002 | Headphone | Electronics | $50 | S002 |

Products Table (Before Insertion)

| **Product ID** | **Name** | **Category** | **Price** | **Supplier ID** |
| --- | --- | --- | --- | --- |
| P003 | Sneakers | Footwear | $80 | NULL |

Attempted Insertion

Issue: Cannot insert 'Sneakers' into the Products table because the Supplier ID is not yet available.

* Delete Anomaly

A delete anomaly happens when the deletion of data inadvertently results in the loss of other valuable data.

Example: Consider an order record that contains crucial customer information. If the order is the only place where a particular customer's information is stored, deleting this order (after completion or cancellation) would also remove the customer’s details from the database. This loss of customer information can be problematic for future marketing efforts, customer service, and analytics.

| **Order ID** | **Customer ID** | **Order Date** | **Total Price** |
| --- | --- | --- | --- |
| O001 | C001 | 2023-11-01 | $100 |
| O002 | C002 | 2023-11-02 | $200 |
| **Order ID** | **Customer ID** | **Order Date** | **Total Price** |

Orders Table (Before Deletion)

| **Customer ID** | **Name** | **Contact Info** |
| --- | --- | --- |
| C001 | John | john@email.com |
| C002 | Emma | emma@email.com |

Customers Table

| **Order ID** | **Customer ID** | **Order Date** | **Total Price** |
| --- | --- | --- | --- |
| O002 | C002 | 2023-11-02 | $200 |

After Deletion of Order ID O001

| **Customer ID** | **Name** | **Contact Info** |
| --- | --- | --- |
| C002 | Emma | emma@email.com |

Customers Table (Post-Deletion)

Issue: Deleting order O001 also removes all records of customer C001 from the database.

* Update Anomaly

An update anomaly occurs when the same data is duplicated in multiple places, and updating it in one place does not automatically update it in others.

Example: Suppose the same product is listed in multiple orders, and there's a need to update its price. If the product price is stored in each order line item rather than just referencing the product table, the price update must be made in multiple places (all order line items containing that product). This is not only time-consuming but also prone to errors, as some instances might get missed, leading to inconsistent data across the database.

| **Order ID** | **Product ID** | **Quantity** | **Price per Item** |
| --- | --- | --- | --- |
| O001 | P001 | 2 | $20 |
| O002 | P001 | 1 | $20 |

Order Details Table (Before Update)

Product Price Update: Product ID P001 price changes from $20 to $25.

| **Order ID** | **Product ID** | **Quantity** | **Price per Item** |
| --- | --- | --- | --- |
| O001 | P001 | 2 | $25 |
| O002 | P001 | 1 | $20 |

Order Details Table (After Inconsistent Update)