**ASSESSMENT 1- (Part B)**

**ICT503 – Database Systems**

**Case Study**

**Name of the Student:**

**Student ID:**

**Introduction**

This document contains the answer to the UPDATE anomaly and the answer to the 3NF normalization for the given abnormal rel schema. This document contains all the details for the tables, that will be created after we have achieved the 3rd normal form. The details includes the tables names, constraints etc. This document also discusses about the different types of issues which can be caused to the data due to the update anomaly.

An UPDATE anomaly is a DBMS issue when the same data is stored redundantly in multiple tables within a database. This redundancy can lead to inconsistencies and anomalies when updating the data. Specifically, an UPDATE anomaly can occur in three ways:

* **Insertion Anomaly:** When you can't add specific data to the database because it depends on other data that hasn't been entered yet. In other words, you can only insert a new record if you provide values for all the attributes, even if they're irrelevant.
* **Deletion Anomaly:** Deleting data inadvertently removes other data you didn't intend to delete because it was linked to the data being removed.
* **Update Anomaly:** When updating one piece of data in a way that causes inconsistencies or loss of data in other related records.

The abnormal rel schema is a mix of attributes related to the tables: products, branches, campaigns, and more. The potential for an UPDATE ANOMALY exists in this schema because:

* **Data Redundancy:** Several attributes can be duplicated across multiple records. Like-ProductType, PackageType, YearProduced, Price, Brand, StockLevel, CampaignStartDate, CampaignEndDate, FirstName, LastName, Email, MembershipLevel, MemberExpDate, and Discount
* **Inconsistent Updates:** Since the data is present in multiple places, updating one record might not be consistent. For example, if we update the price of a product in one place, we might need to remember to update it in another, leading to data consistency.

**Example**:

Let's consider a scenario where multiple products are associated with the same Campaign, and the Campaign's start date is repeated for each product:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ProductId** | **CampaignId** | **CampaignStartDate** | **CampaignEndDate** | **BranchId** |
| 100 | 101 | 28-03-2024 | 28-02-2014 | 1 |
| 101 | 101 | 28-03-2024 | 28-02-2014 | 2 |

Let's say that we have to update the CampaignStartDate for CampaignId 101 and then update for both ProductId 101 and 102. If we forget to update for one product, then it will lead to inconsistency; thus, we can conclude that the schema is susceptible to update anomaly.

**Normalize the schema in 3NF:**

To normalize the schema to 3NF, we will have to follow the following steps given below:

1. **Identifying the functional dependencies:** We must identify the functional dependencies within the original abnormal schema. Based on the attributes provided like:

**ProductID**: ProductType, PackageType, YearProduced, Price, Brand, StockLevel

**BranchID**: CampaignStartDate, CampaignEndDate

**MemberID**: FirstName, LastName, Email, MembershipLevel, MemberExpDate, Discount.

1. **Creation of separate tables:** Based on the functional dependencies identified, we must create separate tables for related attributes. For example:
   1. **Product Table:** 
      1. ProductID (Primary Key)
      2. ProductType
      3. PackageType
      4. YearProduced
      5. Price
      6. Brand
      7. StockLevel
   2. **Campaign Table:** 
      1. BranchID (Primary Key)
      2. CampaignID (Primary Key)
      3. CampaignStartDate
      4. CampaignEndDate
   3. **Member Table:**
      1. MemberID (Primary Key)
      2. FirstName
      3. LastName
      4. Email
      5. MembershipLevel
      6. MemberExpDate
      7. Discount
   4. **Abnormal rel schema table:**
      1. ProductID
      2. BranchID
      3. CampaignID
      4. MemberId
2. **Defining constraints and relationships:**
   1. **Product Table:**

ProductID (Primary Key)

* 1. **Campaign Table:**

BranchID, CampaignID: Primary Key

BranchID: Foreign Key referencing Branch

CampaignID: Foreign Key referencing Campaign

* 1. **Member Table:**

MemberID: Primary Key

* 1. **Abnormal\_Rel Table:**

ProductID: Foreign Key referencing Product

BranchID: Foreign Key referencing Branch

CampaignID: Foreign Key referencing Campaign

MemberID: Foreign Key referencing Member

**VISUALISATION:**

**Product Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ProductID | ProductType | PackageType | YearProduced | Price | Brand | StockLevel |
| 1 | Beer | Bottle | 2024 | 2000 | ABC | 100 |
| 2 | Wine | Bottle | 2024 | 2500 | XYZ | 50 |

**Campaign Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| BranchID | CampaignID | CampaignStartDate | CampaignEndDate |
| 1 | 1 | 2024-03-28 | 2024-04-28 |
| 2 | 2 | 2024-03-28 | 2024-05-28 |

**Abnormal schema table:**

|  |  |  |  |
| --- | --- | --- | --- |
| ProductID | BranchID | CampaignID | MemberID |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 2 |

This way, we decomposed the schema and organized the data into separate tables; thus, we ensured that redundancy was removed and each table was in 3NF form.

**Conclusion**

.So we can say that we have discussed all the issues that can be caused by the update anomaly - like the insertion and delete anomaly. We also have discussed about the 3rd normal form of normalization and have decomposed the abnormal rel schema into different tables and ensured that we have removed the redundancy and each table was in 3NF form.

# **References**