****

**DEPARTMENT OF COMPUTING**

**COMP2350/6350 2021 S2 – ASSIGNMENT TWO (10%)**

***Draft Due: 11:55pm Tuesday 28 September 2021***

**Due: 11:55pm Friday 01 October 2021 (Week 8)**

**Database Design & Manipulation**

Please Print Clearly In **CAPITALS**

|  |  |
| --- | --- |
| **Surname** | **JIA** |
| **First Name** | **RUNDE** |
| **Student ID** | **44434065** |
| **Signature** | **Shape  Description automatically generated with medium confidence** |

#### **Student Code of Conduct**

Macquarie University students have a responsibility to be familiar with the Student Code of Conduct: [https://students.mq.edu.au/study/getting-started/student-conduct​](https://students.mq.edu.au/study/getting-started/student-conduct)

### Student Support

Macquarie University provides a range of support services for students. For details, visit <http://students.mq.edu.au/support/>

**Task 1: Functional Dependencies**

* *Identify the non-trivial FDs on the relation Abnormal\_Rel. Supplement your description with diagram(s).*

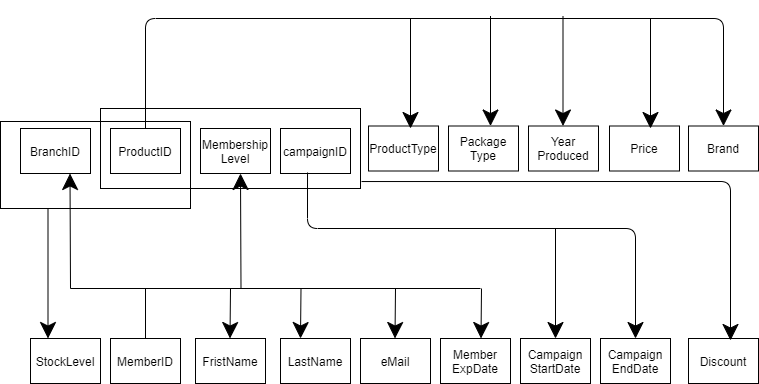
ProductID -> ProductType, PackageType, YearProduced, Price, Brand

ProductID, BranchID -> StockLevel

campaignID -> CampaignStartDate, CampaignEndDate

MemberID -> FirstName, LastName, eMail, MembershipLevel, MemberExpDate, BranchID

ProductID, campaignID, MembershipLevel -> Discount



* *Identify the Candidate key(s) of Abnormal\_Rel.*

(ProductID, BranchID, campaignID, MemberID)

**Task 2: Anomalies**

**Note.** *How you structure your answer is flexible. One possible structure is given below.*

* *Anomalies of Form\_1*
  + *determine if the relation Abnormal\_Rel is susceptible to it*
  + *Support your determination with adequate explanation and a small example (instance of the relation Abnormal\_Rel).*

Relation Abnormal\_Rel is susceptible to insertion anomaly. Insertion anomaly is that when we want to insert an attribute, it depends on other attributes. For example, we cannot insert a new branch that has no product stored yet.

* *Anomalies of Form\_2*
  + *determine if the relation Abnormal\_Rel is susceptible to it*
  + *Support your determination with adequate explanation and a small example (refer to the same or a different instance of the relation Abnormal\_Rel).*

Relation Abnormal\_Rel is susceptible to modification anomaly. We need to update any attribute at multiple locations instead of updating it directly at one place. For example, we need to update the price of one product, we need to change it in a number of places, risking inconsistency due to carelessness.

* *Anomalies of Form\_3*
  + *determine if the relation Abnormal\_Rel is susceptible to it*
  + *Support your determination with adequate explanation and a small example (refer to the same or a different instance of the relation Abnormal\_Rel).*

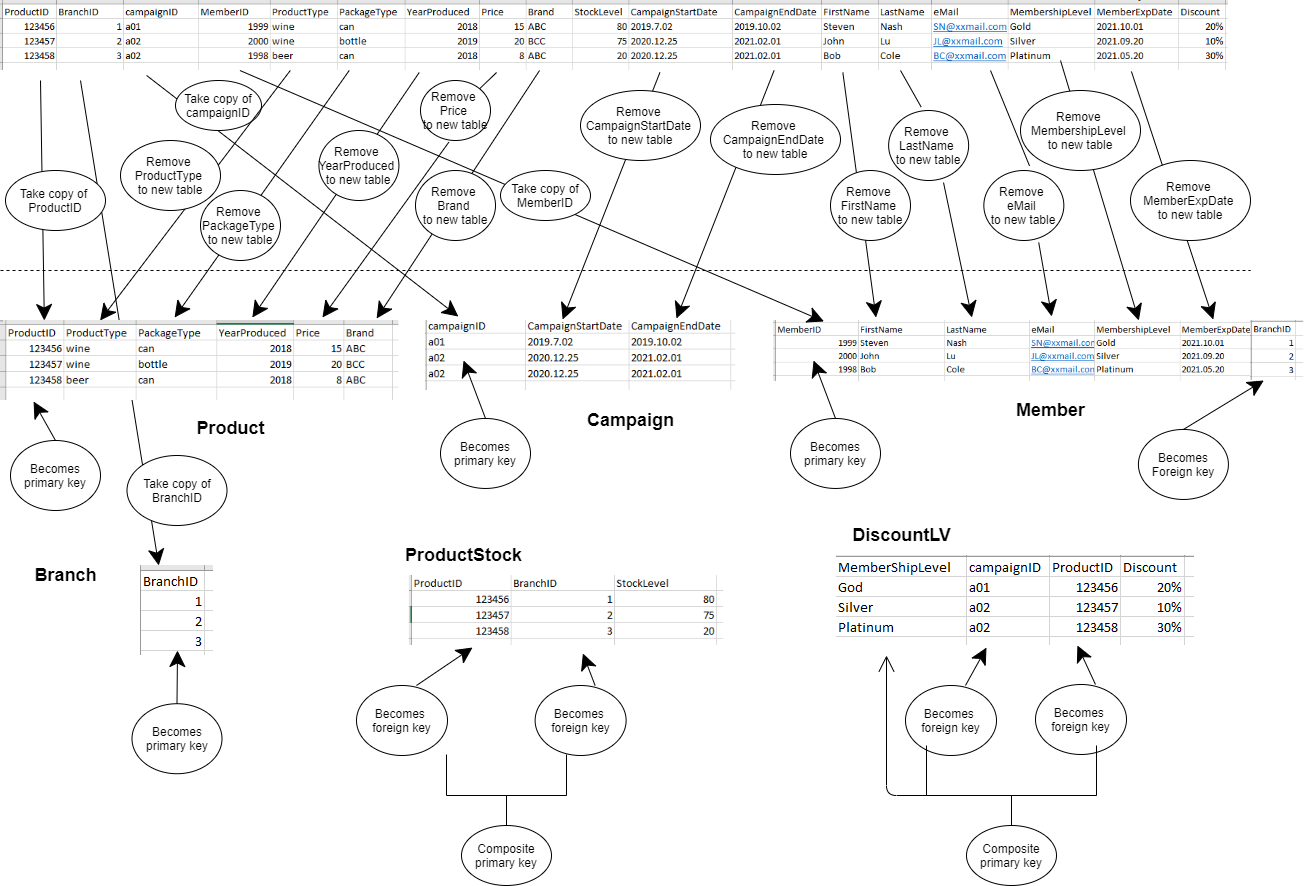
Relation Abnormal\_Rel is susceptible to deletion anomaly. It means if delete an attribute which is not needed it affects in deletion of the other depended attributes which are needed. For example, if one product is out of stock, we want to delete that information. But only one branch is storing that product. Then the whole branch information will be deleted.

**Task 3: Normalization**

* *What is the highest NF that the relation Abnormal\_Rel satisfies? Explain why.*

The relational schema is in FIRST NORMAL FORM, as there is no composite attribute, but there are partial and transitive dependency present in the relation. Several attributes that are dependent only on part of the key, thus violating 2NF.

Values in FirstName, LastName, eMail, MembershipLevel, MemberExpDate column can be worked out from only MemeberID, so table is only in 1NF.

* *Normalize/decompose Abnormal\_Rel until you get relations that are in 3NF. Use appropriate illustration to aid the understanding of your work.*

**Product** (ProductID, ProductType, PackageType, YearProduced, Price, Brand)

**Branch** (BranchID)

**Campaign** (campaignID, CampaignStartDate, CampaignEndDate)

**Member** (MemberID, FirstName, LastName, eMail, MembershipLevel, MemeberExpDate, BranchID)

**ProductStock** (ProductID, BranchID, StockLevel)

**DiscountLV** (MembershipLevel, campaignID, ProductID, Discount)

Product Primary Key: ProductID;

Branch Primary Key: BranchID;

Campaign Composite Primary Key: campaignID;

Member Primary Key: MemberID;

ProductStock Composite Primary Key: (ProductID(Foreign Key), BranchID(Foreign Key))

DiscountLV Composite Primary Key: (MembershipLevel, campaignID (Foreign Key), ProductID(Foreign Key))

After removed partial and transitive dependency present in the relation, the relation now is in 3NF. All attributes are dependent on the whole primary key.

* *Check if the resultant relations are in BCNF. If not, decompose them as necessary until you get all of them in BCNF.*

The primary key is the only determinant (and the only candidate key), so by definition this is in BCNF.

**Task 4: Table Creation and Population**

* *Copy and paste your DDL code for creating each table/relation in BCNF obtained in Task 3.*

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

drop table if exists Product, Branch, Campaign, Member, ProductStock, DiscountLV;

-- -----------------------------------------------------

-- Table `Product`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Product` (

`ProductID` VARCHAR(10) NOT NULL,

`ProductType` VARCHAR(45) NULL,

`PackageType` VARCHAR(45) NULL,

`YearProduced` INT NULL,

`Price` DOUBLE NULL,

`Brand` VARCHAR(45) NULL,

PRIMARY KEY (`ProductID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Campaign`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Campaign` (

`campaignID` VARCHAR(10) NOT NULL,

`CampaignStartDate` DATE NULL,

`CampaignEndDate` DATE NULL,

PRIMARY KEY (`campaignID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Branch`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Branch` (

`BranchID` VARCHAR(10) NOT NULL,

PRIMARY KEY (`BranchID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Member`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Member` (

`MemberID` INT NOT NULL,

`FirstName` VARCHAR(45) NULL,

`LastName` VARCHAR(45) NULL,

`eMail` VARCHAR(45) NULL,

`MembershipLevel` VARCHAR(45) NULL,

`MemberExpDate` DATE NULL,

`BranchID` VARCHAR(10) NOT NULL,

PRIMARY KEY (`MemberID`),

INDEX `fk\_Branch\_has\_Member\_idx` (`BranchID` ASC),

CONSTRAINT `fk\_Branch\_has\_Member`

FOREIGN KEY (`BranchID`)

REFERENCES `Branch` (`BranchID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `ProductStock`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `ProductStock` (

`ProductID` VARCHAR(10) NOT NULL,

`BranchID` VARCHAR(10) NOT NULL,

`StockLevel` INT NULL,

PRIMARY KEY (`BranchID`, `ProductID`),

INDEX `fk\_Branch\_has\_Product\_idx` (`ProductID` ASC),

CONSTRAINT `fk\_Branch\_has\_Product`

FOREIGN KEY (`ProductID`)

REFERENCES `Product` (`ProductID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

INDEX `fk\_Branch\_has\_Product\_idx2` (`BranchID` ASC),

CONSTRAINT `fk\_Branch\_has\_Product2`

FOREIGN KEY (`BranchID`)

REFERENCES `Branch` (`BranchID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `DiscountLV`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `DiscountLV` (

`MembershipLevel` VARCHAR(45) NOT NULL,

`campaignID` VARCHAR(10) NOT NULL,

`ProductID` VARCHAR(10) NOT NULL,

`Discount` VARCHAR(10) NULL,

PRIMARY KEY (`MembershipLevel`, `campaignID`, `ProductID`),

INDEX `fk\_campaignID\_idx` (`campaignID` ASC),

CONSTRAINT `fk\_campaignID`

FOREIGN KEY (`campaignID`)

REFERENCES `Campaign` (`campaignID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

INDEX `fk\_ProductID\_idx` (`ProductID` ASC),

CONSTRAINT `fk\_ProductID`

FOREIGN KEY (`ProductID`)

REFERENCES `Product` (`ProductID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

* *Copy and paste your SQL code for inserting at least five rows of data into each of these table.*

*-- --------------------------------------*

*-- Insert into 'Product'*

*-- --------------------------------------*

*INSERT INTO Product values ('12345', 'wine', 'bottle', '2010', '25.00', 'Penfold Grange');*

*INSERT INTO Product values ('12346', 'beer', 'can', '2019', '8.50', 'Corona Extra');*

*INSERT INTO Product values ('12347', 'beer', 'can', '2018', '10.50', 'Victoria Bitter');*

*INSERT INTO Product values ('12348', 'spirit', 'bottle', '2008', '50.00', 'Absolut Vodaka');*

*INSERT INTO Product values ('12349', 'wine', 'bottle', '2008', '30.00', 'Penfold Grange');*

*-- --------------------------------------*

*-- Insert into 'Branch'*

*-- --------------------------------------*

*INSERT INTO Branch values ('1');*

*INSERT INTO Branch values ('2');*

*INSERT INTO Branch values ('3');*

*INSERT INTO Branch values ('4');*

*INSERT INTO Branch values ('5');*

*-- --------------------------------------*

*-- Insert into 'Campaign'*

*-- --------------------------------------*

*INSERT INTO Campaign values ('a01', '2021-12-10', '2021-12-31');*

*INSERT INTO Campaign values ('a02', '2021-01-01', '2021-05-31');*

*INSERT INTO Campaign values ('a03', '2019-05-05', '2019-07-05');*

*INSERT INTO Campaign values ('a04', '2022-12-10', '2022-12-31');*

*INSERT INTO Campaign values ('a05', '2020-09-30', '2020-10-10');*

*-- --------------------------------------*

*-- Insert into 'Member'*

*-- --------------------------------------*

*INSERT INTO Member values ('1999', 'Nash', 'Steve', 'nashs@xx.mail', 'Gold', '2021-11-05', '1');*

*INSERT INTO Member values ('1998', 'Simone', 'Singh', 'simones@xx.mail', 'Gold', '2021-11-20','2');*

*INSERT INTO Member values ('2000', 'Kiki', 'West', 'kikiw@xx.mail', 'Silver', '2021-12-20','3');*

*INSERT INTO Member values ('1987', 'Dickson', 'Wu', 'dicksonw@xx.mail', 'Platitum', '2021-12-30','1');*

*INSERT INTO Member values ('2001', 'Tom', 'Jerry', 'tomj@xx.mail', 'Silver', '2022-01-05','1');*

*INSERT INTO Member values ('1982', 'Lara', 'Howard', 'larah@xx.mail', 'Gold', '2023-11-05','4');*

*INSERT INTO Member values ('2015', 'Fisher', 'Derrick', 'fisherd@xx.mail', 'Platitum', '2022-05-05','5');*

*-- --------------------------------------*

*-- Insert into 'ProductStock'*

*-- --------------------------------------*

*INSERT INTO ProductStock values ('12345', '1', '10');*

*INSERT INTO ProductStock values ('12345', '2', '30');*

*INSERT INTO ProductStock values ('12347', '1', '8');*

*INSERT INTO ProductStock values ('12348', '3', '100');*

*INSERT INTO ProductStock values ('12349', '2', '1');*

*INSERT INTO ProductStock values ('12346', '4', '10');*

*INSERT INTO ProductStock values ('12346', '5', '50');*

*-- --------------------------------------*

*-- Insert into 'DiscountLV'*

*-- --------------------------------------*

*INSERT INTO DiscountLV values ('Gold', 'a01', '12346', '20%');*

*INSERT INTO DiscountLV values ('Gold', 'a01', '12347', '20%');*

*INSERT INTO DiscountLV values ('Platitum', 'a01', '12348', '40%');*

*INSERT INTO DiscountLV values ('Platitum', 'a02', '12348', '35%');*

*INSERT INTO DiscountLV values ('Silver', 'a01', '12346', '10%');*

*INSERT INTO DiscountLV values ('Silver', 'a05', '12345', '10%');*

* *Copy and paste the SELECT \* query to display the content of each table above, and screenshot of the content as displayed.*

SELECT \* FROM Product;

Table

Description automatically generated

SELECT \* FROM Branch;

Graphical user interface, application, Teams

Description automatically generated

SELECT \* FROM Campaign;

Table

Description automatically generated

SELECT \* FROM Member;

Table

Description automatically generated

SELECT \* FROM ProductStock;

Table

Description automatically generated

SELECT \* FROM DiscountLV;

Table

Description automatically generated

**Task 5: SQL Queries**

Copy and paste the SQL queries followed by their results (screenshot) for each of the following query

**[Query 1]** *List the branches (ID) of MA that have in stock at least 5 bottles of Penfold Grange 2010.*

*SELECT BranchID*

*FROM ProductStock*

*WHERE ProductID = (*

*SELECT ProductID*

*FROM Product*

*WHERE PackageType = 'bottle'*

*AND Brand = 'Penfold Grange'*

*AND YearProduced = '2010'*

*) AND StockLevel >= 5;*

Result:

Table

Description automatically generated with low confidence

**[Query 2]** *List details of each beer that Simone Singh will be entitled to get 20% discount on.*

SELECT \*

FROM Product

WHERE ProductID IN (

SELECT ProductID

FROM DiscountLV

WHERE Discount = '20%'

AND MembershipLevel = (

SELECT MembershipLevel

FROM Member

WHERE FirstName = 'Simone'

AND LastName = 'Singh'

)

AND campaignID = (

SELECT campaignID

FROM Campaign

WHERE '2021-12-24' BETWEEN CampaignStartDate AND CampaignEndDate

)

);

Result:

Table

Description automatically generated

**[Query 3]** *Generate a list of all email addresses of members whose card will expire in the month after the coming month, ordered appropriately.*

SELECT eMail

FROM Member

WHERE MemberExpDate BETWEEN DATE\_SUB(

LAST\_DAY(

DATE\_ADD(NOW(), INTERVAL 2 MONTH)

),

INTERVAL DAY(

LAST\_DAY(

DATE\_ADD(NOW(), INTERVAL 2 MONTH)

)

)-1 DAY) AND LAST\_DAY(

DATE\_ADD(NOW(), INTERVAL 2 MONTH)

)

ORDER BY BranchID ASC,

MemberExpDate ASC,

eMail ASC;

Result:

Graphical user interface, text, application

Description automatically generated

**[Query 4]** *Determine how many times Penfold Grange 2010 has gone on sale since Covid-19 related lockdown started (assume it to be March 01, 2020).*

SELECT COUNT(\*)

FROM DiscountLV

WHERE ProductID = (

SELECT ProductID

FROM Product

WHERE Brand = 'Penfold Grange' AND YearProduced = '2010'

)

AND campaignID in (

SELECT campaignID

FROM Campaign

WHERE (CampaignEndDate BETWEEN '2020-03-01' AND curdate())

OR (CampaignStartDate BETWEEN '2020-03-01' AND curdate())

);

**Result:**

**Graphical user interface

Description automatically generated with low confidence**