**General Constraints**

**Constraint 1:**

A brand can have one of two types of loyalty programs: tiered or regular. This is ensured by the DBMS by having a LP table with the primary key being an individual brand's unique brand id. This means each brand can only have one entry in the LP table therefore one loyalty program.

CREATE TABLE PKANAKI.LP

(

LOYALTYPROGRAMCODEID VARCHAR2(6) NOT NULL ,

NAME VARCHAR2(100) NOT NULL ,

STATE VARCHAR2(100) NOT NULL ,

BRANDID VARCHAR2(100) NOT NULL UNIQUE,

TIER NUMBER(2,0) NOT NULL,

PRIMARY KEY (LOYALTYPROGRAMCODEID),

FOREIGN KEY (BRANDID) REFERENCES PKANAKI.BRAND(BRANDID)

);

**Constraint 2:**

A tiered loyalty program has a minimum of 2 tiers and a maximum of 3 tiers. This has to be enforced in the application since check constraints in oracle db 12c can not contain subqueries. We would need a subquery to select and count the number of tiers in the Tier table to ensure a Tiered Loyalty Program for a brand did not have less than 2 or greater than 3 tiers. An example of how we tried to enforce this constraint in the dbms.

Loyalty Program:

// A Tiered Loyalty program can not have less than 2 tiers (can’t use because no subqueries)

**ALTER** **TABLE** PKANAKI.LP **ADD** (**CONSTRAINT** check\_tiers

**CHECK** ((

**SELECT** t.BRANDID, **COUNT**(\*)

**FROM** PKANAKI.TIER t, PKANAKI.LP l

**WHERE** t.BRANDID = l.BRANDID **AND** l.TIER = 1

**GROUP** **BY** t.BRANDID

**HAVING** **COUNT**(\*) < 2) = 0));

TIERS:

// A Tiered Loyalty program can not have more than 3 tiers (can’t use because no subqueries)

**ALTER** **TABLE** PKANAKI.TIER **ADD** (**CONSTRAINT** check\_tiers

**CHECK** ((

**SELECT** t.BRANDID, **COUNT**(\*)

**FROM** PKANAKI.TIER t

**GROUP** **BY** t.BRANDID

**HAVING** **COUNT**(\*) > 3) = 0));

**Constraint 3:**

Tiers have to have precedence ordering. This has to be enforced in the application since check constraints in oracle db 12c can not contain subqueries.

**Constraint 4:**

Loyalty Programs must have 1 RE Rule and 1 RR Rule. This has to be enforced in the application since check constraints in oracle db 12c can not contain subqueries. We would need a subquery to check the number of RE Rules for a brand and RR Rules for a brand on insertion/update to the Loyalty Program table to update the state to active. Also oracle db 12c does not allow referencing other tables in check constraints so we would not be able to reference the Tier table on a check constraint for the validation of a Loyalty Program in the LP table.

**Constraint 5**

Customers can choose which brand’s loyalty programs they want to be enrolled in. Thus the ENROLLED table contains CUSTOMERID, BRANDID as the primary key. This table is referencing CUSTOMERID from CUSTOMER table and BRANDID from LP Table.

CREATE TABLE PKANAKI.ENROLLED

(

CUSTOMERID VARCHAR2(100) NOT NULL,

BRANDID VARCHAR2(100) NOT NULL,

"NAME" VARCHAR2(100) NOT NULL,

PRIMARY KEY (CUSTOMERID,BRANDID),

FOREIGN KEY (CUSTOMERID) REFERENCES PKANAKI.CUSTOMER(CUSTOMERID),

FOREIGN KEY (BRANDID) REFERENCES PKANAKI.LP(BRANDID)

);

**Constraint 6**

Each customer has a wallet that is associated with the brand’s loyalty program they are enrolled in. Thus, the WALLET table contains CUSTOMERID, BRANDID as the primary key and it references CUSTOMERID, BRANDID from ENROLLED Table.

CREATE TABLE PKANAKI.WALLET

(

CUSTOMERID VARCHAR2(100) NOT NULL,

BRANDID VARCHAR2(100) NOT NULL,

POINTSEARNED NUMBER DEFAULT 0,

POINTSREDEEMED NUMBER DEFAULT 0,

TIER NUMBER DEFAULT 0,

MULTIPLIER NUMBER DEFAULT 1,

PRIMARY KEY (CUSTOMERID,BRANDID),

FOREIGN KEY (CUSTOMERID,BRANDID) REFERENCES PKANAKI.ENROLLED(CUSTOMERID,BRANDID)

);

**Constraint 7**

A reward activity occurs when a customer redeems points that were earned from a loyalty program. Therefore, each RewardActivity is associated with a Brand, a Customer and a RewardCategoryCode. Because customers can perform the same reward activity, date must be included in the key so we can distinguish each RewardActivity. The primary key for REWARDACTIVITY is CUSTOMERID, BRANDID, REWARDCATEGORYCODE, and REWARDDATE. It references CUSTOMERID,BRANDID from the ENROLLED table and BRANDID,REWARDCATEGORYCODE,VERSION from the RRRULETABLE table.

CREATE TABLE PKANAKI.REWARDACTIVITY

(

CUSTOMERID VARCHAR2(100) NOT NULL,

BRANDID VARCHAR2(100) NOT NULL,

REWARDCATEGORYCODE VARCHAR2(6) NOT NULL,

REWARDDATE DATE NOT NULL,

POINTSREDEEMED NUMBER NOT NULL,

VERSION NUMBER NOT NULL,

PRIMARY KEY (CUSTOMERID,BRANDID,REWARDCATEGORYCODE,REWARDDATE),

FOREIGN KEY (CUSTOMERID,BRANDID) REFERENCES PKANAKI.ENROLLED(CUSTOMERID,BRANDID),

FOREIGN KEY (BRANDID,REWARDCATEGORYCODE,VERSION) REFERENCES PKANAKI.RRRULETABLE(BRANDID,REWARDCATEGORYCODE,VERSION)

);

**Constraint 8**

A loyalty activity occurs when a customer performs a specific action that a brand has in order to earn points. This means that each LoyaltyActivity is associated with a Brand, Customer, and ActivityCategoryCode. The date of the activity also must be a part of the key as the same activity can be performed multiple times by a customer. Therefore, CUSTOMERID, BRANDID, ACTIVITYCATEGORYCODE, ACTIVITYDATE is the primary key for the LOYALTYACTIVITY table. It references CUSTOMERID, BRANDID from the ENROLLED table and BRANDID, ACTIVITYCATEGORYCODE, VERSION from the RERULETABLE.

CREATE TABLE PKANAKI.LOYALTYACTIVITY

(

CUSTOMERID VARCHAR2(100) NOT NULL,

BRANDID VARCHAR2(100) NOT NULL,

ACTIVITYCATEGORYCODE VARCHAR2(6) NOT NULL,

ACTIVITYDATE DATE NOT NULL,

POINTSEARNED NUMBER NOT NULL,

VERSION NUMBER NOT NULL,

PRIMARY KEY (CUSTOMERID,BRANDID,ACTIVITYCATEGORYCODE,ACTIVITYDATE),

FOREIGN KEY (CUSTOMERID,BRANDID) REFERENCES PKANAKI.ENROLLED(CUSTOMERID,BRANDID),

FOREIGN KEY (BRANDID,ACTIVITYCATEGORYCODE,VERSION) REFERENCES PKANAKI.RERULETABLE(BRANDID,ACTIVITYCATEGORYCODE,VERSION)

);

**Constraint 9**

A brand can choose to have a loyalty program with tiers. If they do have a tiered program, each Tier will be stored in this table. Thus BrandID is foreign key in the Tier table. BRANDID and TIERLEVEL are the primary key since brands may have multiple tiers.

CREATE TABLE PKANAKI.TIER

(

BRANDID VARCHAR2(100) NOT NULL,

TIERLEVEL NUMBER NOT NULL,

TIERNAME VARCHAR2(100) NOT NULL,

POINTSREQUIRED NUMBER NOT NULL,

MULTIPLIER NUMBER DEFAULT 1 NOT NULL,

PRIMARY KEY (BRANDID,TIERLEVEL),

FOREIGN KEY (BRANDID) REFERENCES PKANAKI.LP(BRANDID)

);

**Constraint 10**

A brand can choose the different types of activities they will use in their loyalty program. Thus, the BRAND\_ACTIVITYTYPE table contains BRANDID, ACTIVITYCATEGORYCODE as the primary key with BrandID referencing the BRAND table and ACTIVITYCATEGORYCODE referencing the ACTIVITYCATEGORY table.

CREATE TABLE PKANAKI.BRAND\_ACTIVITYTYPE

(

BRANDID VARCHAR2(100) NOT NULL,

ACTIVITYCATEGORYCODE VARCHAR2(6) NOT NULL,

PRIMARY KEY (BRANDID,ACTIVITYCATEGORYCODE),

FOREIGN KEY (BRANDID) REFERENCES PKANAKI.LP(BRANDID),

FOREIGN KEY (ACTIVITYCATEGORYCODE) REFERENCES PKANAKI.ACTIVITYCATEGORY(ACTIVITYCATEGORYCODE)

);

**Constraint 11**

A brand can generate different types of rewards for their loyalty program. The BRAND\_REWARDTYPE table has the primary key BRANDID, REWARDCATEGORYCODE. This way each brand can choose which types of reward they would like in their program. The BRAND\_REWARDTYPE table references BRANDID from the BRAND table and REWARDCATEGORYCODE from the REWARDCATEGORY Table.

CREATE TABLE PKANAKI.BRAND\_REWARDTYPE

(

BRANDID VARCHAR2(100) NOT NULL,

REWARDCATEGORYCODE VARCHAR2(6) NOT NULL,

INSTANCES NUMBER NOT NULL CHECK(INSTANCES >= 0),

PRIMARY KEY (BRANDID,REWARDCATEGORYCODE),

FOREIGN KEY (BRANDID) REFERENCES PKANAKI.LP(BRANDID),

FOREIGN KEY (REWARDCATEGORYCODE) REFERENCES PKANAKI.REWARDCATEGORY(REWARDCATEGORYCODE)

);

**Constraint 12**

Each brand can create reward earning rules for their loyalty program. The rule will specify how many points a customer will receive if they perform a specific activity. The number of points earned can be different for each activity. Earning points are dependent on a combination of BRANDID and REWARDCATEGORYCODE. A brand can also choose to update their rules so a version number must be a part of the key. Thus the primary key is BRANDID, ACTIVITYCATEGORYCODE, VERSION and it references the ACTIVITYTYPE table.

CREATE TABLE PKANAKI.RERULETABLE

(

BRANDID VARCHAR2(100) NOT NULL,

ACTIVITYCATEGORYCODE VARCHAR2(6) NOT NULL,

POINTS NUMBER DEFAULT 0 NOT NULL,

VERSION NUMBER DEFAULT 1 NOT NULL,

PRIMARY KEY (BRANDID,ACTIVITYCATEGORYCODE,VERSION),

FOREIGN KEY (BRANDID,ACTIVITYCATEGORYCODE) REFERENCES PKANAKI.BRAND\_ACTIVITYTYPE(BRANDID,ACTIVITYCATEGORYCODE)

);

**Constraint 13**

Each brand can create reward redeeming rules for their loyalty program. The rule will specify how many points a customer needs to receive a reward from the brand. The number of points needed for each reward could be different. Earning points are dependent on a combination of BRANDID and REWARDCATEGORYCODE. A brand can also choose to update their rules so a version number must be a part of the key. Thus the primary key is BRANDID, ACTIVITYCATEGORYCODE, VERSION and it references the ACTIVITYTYPE table.

CREATE TABLE PKANAKI.RRRULETABLE

(

BRANDID VARCHAR2(100) NOT NULL,

REWARDCATEGORYCODE VARCHAR2(6) NOT NULL,

POINTS NUMBER DEFAULT 0 NOT NULL,

INSTANCES NUMBER NOT NULL CHECK(INSTANCES >= 0),

VERSION NUMBER DEFAULT 1 NOT NULL,

PRIMARY KEY (BRANDID,REWARDCATEGORYCODE,VERSION),

FOREIGN KEY (BRANDID,REWARDCATEGORYCODE) REFERENCES PKANAKI.BRAND\_REWARDTYPE(BRANDID,REWARDCATEGORYCODE)

);

**Constraints enforced by Procedures**

**Procedure 1:**

When a customer performs a loyalty activity for a brand, the number of points they earn for the activity depends on the RE rule and the multiplier of the tier the customer is in for the brand (for tiered loyalty programs only). The correct version of the RE rule must also be selected. The procedure will take care of finding the multiplier, the points earned from the specific activity, and the most current version of the rule. Because this specific task will be repeated many times in the application, it was best to create a procedure for this.

CREATE OR REPLACE PROCEDURE PKANAKI.INSERT\_LOYALTYACTIVITY

(

INP\_CUSTOMERID VARCHAR2,

INP\_BRANDID VARCHAR2,

INP\_ACTIVITYCATEGORYCODE VARCHAR2

)

IS

isvalid NUMBER;

P NUMBER;

M NUMBER;

V NUMBER;

BEGIN

SELECT COALESCE(MULTIPLIER,1) INTO M FROM PKANAKI.WALLET T1 WHERE T1.CUSTOMERID = INP\_CUSTOMERID AND T1.BRANDID = INP\_BRANDID;

SELECT COALESCE(POINTS,0)\*M INTO P FROM PKANAKI.RERULETABLE T1 WHERE T1.BRANDID = INP\_BRANDID AND T1.ACTIVITYCATEGORYCODE = INP\_ACTIVITYCATEGORYCODE ORDER BY VERSION DESC FETCH NEXT 1 ROWS ONLY;

SELECT COALESCE(VERSION,1) INTO V FROM PKANAKI.RERULETABLE T1 WHERE T1.BRANDID = INP\_BRANDID AND T1.ACTIVITYCATEGORYCODE = INP\_ACTIVITYCATEGORYCODE ORDER BY VERSION DESC FETCH NEXT 1 ROWS ONLY;

INSERT INTO PKANAKI.LOYALTYACTIVITY (CUSTOMERID, BRANDID, ACTIVITYCATEGORYCODE, ACTIVITYDATE, POINTSEARNED,VERSION) VALUES(INP\_CUSTOMERID,INP\_BRANDID , INP\_ACTIVITYCATEGORYCODE, SYSDATE, P,V);

END;

**Procedure 2:**

A customer can redeem the points they’ve earned from a brand’s loyalty program for a type of reward. We must check if they have enough points to perform the action, make sure the brand has the RR rule, and find the correct version of the RR rule to use. In addition, we need to update the customer’s wallet to reflect their new point value and decrease the number of rewards that the brand has available to customers. This task has many different checks associated with it that involve querying the tables. Therefore, the logic was best implemented as a procedure.

CREATE OR REPLACE PROCEDURE PKANAKI.INSERT\_REWARDACTIVITY

(

INP\_CUSTOMERID VARCHAR2,

INP\_BRANDID VARCHAR2,

INP\_REWARDCATEGORYCODE VARCHAR2,

INP\_INSTANCES NUMBER

)

IS

isvalid NUMBER;

points\_needed NUMBER;

points\_available NUMBER;

V NUMBER;

BEGIN

SELECT INSTANCES into isvalid FROM PKANAKI.RRRULETABLE T1 WHERE T1.BRANDID = INP\_BRANDID AND REWARDCATEGORYCODE = INP\_REWARDCATEGORYCODE ORDER BY VERSION DESC FETCH NEXT 1 ROWS ONLY;

IF isvalid - INP\_INSTANCES >= 0 THEN

SELECT COALESCE(POINTS,0)\*INP\_INSTANCES INTO points\_needed FROM PKANAKI.RRRULETABLE T1 WHERE T1.BRANDID = INP\_BRANDID AND T1.REWARDCATEGORYCODE = INP\_REWARDCATEGORYCODE ORDER BY VERSION DESC FETCH NEXT 1 ROWS ONLY;

SELECT (POINTSEARNED-POINTSREDEEMED) INTO points\_available FROM PKANAKI.WALLET WHERE BRANDID = INP\_BRANDID AND CUSTOMERID = INP\_CUSTOMERID ;

IF points\_available >= points\_needed THEN

SELECT VERSION INTO V FROM PKANAKI.RRRULETABLE T1 WHERE T1.BRANDID = INP\_BRANDID AND T1.REWARDCATEGORYCODE = INP\_REWARDCATEGORYCODE ORDER BY VERSION DESC FETCH NEXT 1 ROWS ONLY;

INSERT INTO PKANAKI.REWARDACTIVITY (CUSTOMERID, BRANDID, REWARDCATEGORYCODE, REWARDDATE, POINTSREDEEMED,VERSION) VALUES(INP\_CUSTOMERID,INP\_BRANDID , INP\_REWARDCATEGORYCODE, SYSDATE, points\_needed,V);

UPDATE PKANAKI.WALLET SET POINTSREDEEMED = POINTSREDEEMED + points\_needed WHERE BRANDID = INP\_BRANDID AND CUSTOMERID = INP\_CUSTOMERID ;

UPDATE PKANAKI.RRRULETABLE SET INSTANCES = INSTANCES - INP\_INSTANCES WHERE BRANDID = INP\_BRANDID AND REWARDCATEGORYCODE = INP\_REWARDCATEGORYCODE AND VERSION = V;

END IF;

END IF;

END;

**Functions**

**Function 1:**

The application allows any type of user to enter their username and password. If the user exists in the system, they will automatically be taken to the appropriate user landing page such as the Brand landing page, the Customer landing page, or the Admin landing page. This required us to create a function so that a user’s credentials could be validated and also tell what type of user they are by querying each user table.

CREATE OR REPLACE FUNCTION PKANAKI.validateLogin(

INP\_USERID VARCHAR2,

INP\_PASSWORD VARCHAR2

)

RETURN VARCHAR2

IS

isvalid number;

BEGIN

SELECT COUNT(\*) INTO isvalid FROM PKANAKI.CUSTOMER WHERE CUSTOMERID = INP\_USERID AND password = INP\_PASSWORD;

IF isvalid > 0 THEN

RETURN ('1');

END IF;

SELECT COUNT(\*) AS t INTO isvalid FROM PKANAKI.BRAND WHERE BRANDID = INP\_USERID AND password = INP\_PASSWORD;

IF isvalid > 0 THEN

RETURN ('2');

END IF;

SELECT COUNT(\*) AS t INTO isvalid FROM PKANAKI.ADMIN WHERE ADMINID = INP\_USERID AND password = INP\_PASSWORD;

IF isvalid > 0 THEN

RETURN ('3');

END IF;

RETURN ('0');

END validateLogin;

**Function 2:**

After a brand has created a loyalty program and entered any information, rules, reward types, and reward activities, the loyalty program is labeled inactive. In order to activate the loyalty program, it must have at least one RR rule, one RE rule, and if the program is tiered, it must have either 2 or 3 tiers. This function will check all three of those constraints and if the loyalty program is valid, it will then be marked as active.

CREATE OR REPLACE FUNCTION PKANAKI.validateLP(

INP\_BRANDID VARCHAR2

)

RETURN VARCHAR2

IS

isvalid number;

isvalid\_tier NUMBER;

BEGIN

SELECT COUNT(\*) INTO isvalid FROM PKANAKI.RERULETABLE WHERE BRANDID = INP\_BRANDID;

IF isvalid <= 0 THEN

RETURN ('1');

END IF;

SELECT COUNT(\*) INTO isvalid FROM PKANAKI.RRRULETABLE WHERE BRANDID = INP\_BRANDID ;

IF isvalid <= 0 THEN

RETURN ('2');

END IF;

SELECT COUNT(\*) INTO isvalid FROM PKANAKI.LP WHERE BRANDID = INP\_BRANDID AND TIER = 1;

IF isvalid > 0 THEN

SELECT COUNT(\*) INTO isvalid\_tier FROM PKANAKI.TIER WHERE BRANDID = INP\_BRANDID;

IF isvalid\_tier <2 THEN

RETURN ('3');

END IF;

END IF;

RETURN ('0');

END validateLP;

**Triggers**

**Trigger 1:**

A customer will have a wallet for each brand’s loyalty program they are enrolled in. This way a customer can see the programs they are enrolled in and see how many points they’ve earned. Therefore, each time a customer enrolls in a program, a wallet must be created. Because this happens each time a customer enrolls in a loyalty program, it is best implemented as a trigger. After they sign up, a wallet will automatically be created for them based on the brand’s loyalty program.

CREATE OR REPLACE TRIGGER PKANAKI.SETWALLET

AFTER INSERT

ON PKANAKI.ENROLLED

FOR EACH ROW

DECLARE

isvalid number;

M number;

BEGIN

SELECT count(\*) INTO isvalid FROM PKANAKI.LP T1 WHERE TIER=0 AND T1.BRANDID = :new.BRANDID;

IF isvalid>0 THEN

INSERT INTO PKANAKI.WALLET(CUSTOMERID,BRANDID,POINTSEARNED,POINTSREDEEMED,TIER,MULTIPLIER) VALUES(:new.CUSTOMERID,:new.BRANDID,0,0,0,1);

END IF;

SELECT count(\*) INTO isvalid FROM PKANAKI.LP T1 WHERE TIER=1 AND T1.BRANDID = :new.BRANDID;

IF isvalid>0 THEN

SELECT COALESCE(MULTIPLIER,1) INTO M FROM PKANAKI.TIER T1 WHERE T1.BRANDID = :new.BRANDID AND T1.TIERLEVEL = 1;

INSERT INTO PKANAKI.WALLET(CUSTOMERID,BRANDID,POINTSEARNED,POINTSREDEEMED,TIER,MULTIPLIER) VALUES(:new.CUSTOMERID,:new.BRANDID,0,0,1,M) ;

END IF;

END;

**Trigger 2:**

After a customer has performed a loyalty activity the wallet of the customer needs to be updated. The wallet POINTSEARNED field is updated using the multiplier for the tier the customer is at for the specific brand and the points earned for the specific activity performed. This also updates the TIER and MULTIPLIER of the customer based on the number of total points earned by the customer.

/\*TRIGGER UPDATE WALLET AFTER LOYALTYACTIVITY\*/

CREATE OR REPLACE TRIGGER PKANAKI.UPDATEWALLET

AFTER INSERT

ON PKANAKI.LOYALTYACTIVITY

FOR EACH ROW

DECLARE

isvalid number;

latestpoints NUMBER;

M NUMBER;

TIERL NUMBER;

BEGIN

UPDATE PKANAKI.WALLET SET POINTSEARNED = POINTSEARNED + :NEW.POINTSEARNED WHERE CUSTOMERID = :NEW.CUSTOMERID AND BRANDID = :NEW.BRANDID;

SELECT COUNT(\*) INTO isvalid FROM PKANAKI.LP T1 WHERE T1.TIER = '1' AND T1.BRANDID = :NEW.BRANDID;

IF isvalid > 0 THEN

SELECT POINTSEARNED INTO latestpoints FROM PKANAKI.WALLET WHERE CUSTOMERID = :NEW.CUSTOMERID AND BRANDID = :NEW.BRANDID;

SELECT MULTIPLIER INTO M FROM PKANAKI.TIER T1 WHERE T1.BRANDID=:NEW.BRANDID AND latestpoints >= T1.POINTSREQUIRED ORDER BY TIERLEVEL DESC FETCH NEXT 1 ROWS ONLY;

SELECT TIERLEVEL INTO TIERL FROM PKANAKI.TIER T1 WHERE T1.BRANDID=:NEW.BRANDID AND latestpoints >= T1.POINTSREQUIRED ORDER BY TIERLEVEL DESC FETCH NEXT 1 ROWS ONLY;

UPDATE PKANAKI.WALLET SET MULTIPLIER = M, TIER = TIERL WHERE CUSTOMERID = :NEW.CUSTOMERID AND BRANDID = :NEW.BRANDID;

END IF;

END;