```
-- The following trigger checks before the insertion or the update of the Card relation
-- if only one card associated to the customer related to the modification is enable.

CREATE FUNCTION mbt.checkEnabledCard() RETURNS TRIGGER AS $$
    BEGIN
        IF(NEW.enabled = TRUE) THEN
            -- Check if there is another card enabled with same customer_id
            PERFORM customer
                FROM mbt.Card
                WHERE customer = NEW.customer AND card_id != NEW.card_id AND enabled = TRUE;
            IF FOUND THEN
                RAISE EXCEPTION 'Customer % already has an enabled card.', new.customer;
            END IF;
            RETURN NEW;
        ELSE
            -- Check if there is another card enabled with same customer_id
            PERFORM customer
                FROM mbt.Card
                WHERE customer = NEW.customer AND enabled = TRUE;
            IF FOUND THEN RETURN NEW;
            ELSE
                NEW.enabled = TRUE;
                RETURN NEW;
            END IF;
        END IF;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER checkCard
    BEFORE INSERT OR UPDATE ON mbt.Card
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.checkEnabledCard();


-- This trigger checks if the value of the attribute Type of the relation Subscription is
consistent
-- with the duration type specified, in case of “daily” subscriptions Hours_duration must be
NULL and Days_duration NOT NULL,
-- viceversa in case of “hourly” subscriptions the former must be NOT NULL and the latter
NULL.

CREATE FUNCTION mbt.checkSubTypeDuration() RETURNS TRIGGER AS $$
    BEGIN
        IF (NEW.type = 'daily') THEN
            IF (NEW.days_duration IS NOT NULL AND NEW.hours_duration IS NULL) THEN
                RETURN NEW;
            ELSE
                RAISE EXCEPTION 'Subscriptions of type "daily" must have days_duration NOT
                NULL and hours_duration NULL';
            END IF;
        ELSIF (NEW.type = 'hourly') THEN
            IF (NEW.days_duration IS NULL AND NEW.hours_duration IS NOT NULL) THEN
                RETURN NEW;
            ELSE
                RAISE EXCEPTION 'Subscriptions of type "hourly" must have days_duration NULL
                and hours_duration NOT NULL';
            END IF;
        END IF;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER checkSubscription
    BEFORE INSERT OR UPDATE ON mbt.Subscription
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.checkSubTypeDuration();


-- This couple of triggers upon insertion or deletion of a docking point updates the count
```

```
-- of the column number in the respective docking station.

CREATE FUNCTION mbt.increaseDockingCount() RETURNS TRIGGER AS $$
    BEGIN
        UPDATE mbt.dockingstation SET columns_number = columns_number + 1
            WHERE docking_station_id = NEW.docking_station;
        RETURN NEW;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER insertedDockPoint
    AFTER INSERT ON mbt.dockingpoint
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.increaseDockingCount();

CREATE FUNCTION mbt.decreaseDockingCount() RETURNS TRIGGER AS $$
    BEGIN
        UPDATE mbt.dockingstation SET columns_number = columns_number - 1
            WHERE docking_station_id = OLD.docking_station;
        RETURN OLD;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER deletedDockPoint
    BEFORE DELETE ON mbt.dockingpoint
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.decreaseDockingCount();


-- Update mbt.Card.current_credit after a credit action

CREATE FUNCTION mbt.updateCurrentCredit() RETURNS TRIGGER AS $$
    BEGIN
        IF (NEW.hire IS NOT NULL) THEN
            -- decrease mbt.Card.current_credit of NEW.value
            UPDATE mbt.Card SET current_credit = (current_credit - NEW.value)
                WHERE card_id = NEW.card AND organization = NEW.organization;
            RETURN NEW;
        ELSIF(NEW.charge IS NOT NULL) THEN
            -- increase mbt.Card.current_credit of NEW.value
            UPDATE mbt.Card SET current_credit = (current_credit + NEW.value)
                WHERE card_id = NEW.card AND organization = NEW.organization;
            RETURN NEW;
        END IF;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER updateCredit
    BEFORE INSERT ON mbt.CreditAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.updateCurrentCredit();

--Query Test

-- Insert a Payment for a subscription
INSERT INTO mbt.charge (transaction_id, date, payment_method) VALUES
('f2bd45b9-c740-4df7-9fe2-589700939a04', CURRENT_DATE, 1);

-- Insert a credit action for increase money in the Card.
INSERT INTO mbt.creditaction (value, card, organization, charge, hire) VALUES (5,
'1456896563', 2, 'f2bd45b9-c740-4df7-9fe2-589700939a04', NULL)

-- Insert a credit action for decrease money in the card
INSERT INTO mbt.creditaction (value, card, organization, charge, hire) VALUES (2,
'1456896563', 2, NULL, 1)
-- Result: OK
INSERT INTO mbt.creditaction (value, card, organization, charge, hire) VALUES (5,
'1456896563', 2, NULL, 1)
```

```
-- Result: OK -> negative current_credit.

--END Query Test


-- Update mbt.Card.current_points afetr a points action

CREATE FUNCTION mbt.updateCurrentPoints() RETURNS TRIGGER AS $$
    DECLARE
        cardPoints INT;
    BEGIN
        IF (NEW.hire IS NOT NULL) THEN
            -- decrease mbt.Card.current_points of NEW.value
            UPDATE mbt.Card SET current_points = (current_points + NEW.value)
                WHERE card_id = NEW.card AND organization = NEW.organization;
            RETURN NEW;
        ELSIF(NEW.offer IS NOT NULL) THEN
            -- Select current_points in the user's card
            SELECT current_points INTO cardPoints
                FROM mbt.Card
                WHERE card_id = NEW.card AND organization = NEW.organization;
            IF (cardPoints > NEW.value) THEN
                -- increase mbt.Card.current_points of NEW.value
                UPDATE mbt.Card SET current_points = (current_points - NEW.value)
                    WHERE card_id = NEW.card AND organization = NEW.organization;
                RETURN NEW;
            ELSE
                RAISE EXCEPTION 'Card''s points must be greater than offer''s cost.';
            END IF;
        END IF;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER updatePoints
    BEFORE INSERT ON mbt.PointsAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.updateCurrentPoints();

-- Query Test

INSERT INTO mbt.pointsaction (value, card, organization, offer, hire) VALUES (50,
'1456896563', 2, NULL, 1);

INSERT INTO mbt.pointsaction (value, card, organization, offer, hire) VALUES (100,
'1456896563', 2, 1, NULL);
-- Resutl: RAISE EXCEPTION

INSERT INTO mbt.pointsaction (value, card, organization, offer, hire) VALUES (60,
'1456896563', 2, NULL, 1);

INSERT INTO mbt.pointsaction (value, card, organization, offer, hire) VALUES (100,
'1456896563', 2, 1, NULL);
-- Result OK

--END Query Test


-- Check if current_credit in User's Card is positive

CREATE FUNCTION mbt.checkCurrentCredit() RETURNS TRIGGER AS $$
    DECLARE
        credit NUMERIC;
    BEGIN
        SELECT current_credit INTO credit
            FROM mbt.Card
            WHERE card_id = NEW.card AND organization = NEW.organization;
        IF (credit > 0) THEN
            RETURN NEW;
```

```
        ELSE
            RAISE EXCEPTION 'Card''s credit must be positive.';
        END IF;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER checkCreditHire
    BEFORE INSERT OR UPDATE ON mbt.Hire
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.checkCurrentCredit();

CREATE TRIGGER checkCreditBooking
    BEFORE INSERT OR UPDATE ON mbt.BookingAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.checkCurrentCredit();

-- Query Test

INSERT INTO mbt.Hire (card, organization, bike, docking_point_unlock, date_unlock) VALUES
('1456896563', 2, 2, 8, '2018-05-20 08:22:54+02');
-- Return Raise because current_credit is negative.


-- Check if used Card is enable

CREATE FUNCTION mbt.isCardEnabled() RETURNS TRIGGER AS $$
    DECLARE
        isEnabled BOOLEAN;
    BEGIN
        SELECT enabled INTO isEnabled
            FROM mbt.Card
            WHERE card_id = NEW.card AND organization = NEW.organization;
        IF (isEnabled) THEN
            RETURN NEW;
        ELSE
            RAISE EXCEPTION 'User''s Card must be enabled.';
        END IF;
    END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER enabledCardSubscription
    BEFORE INSERT OR UPDATE ON mbt.SubscriptionAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.isCardEnabled();

CREATE TRIGGER enabledCardHire
    BEFORE INSERT OR UPDATE ON mbt.Hire
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.isCardEnabled();

CREATE TRIGGER enabledCardPoints
    BEFORE INSERT OR UPDATE ON mbt.PointsAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.isCardEnabled();

CREATE TRIGGER enabledCardCredit
    BEFORE INSERT OR UPDATE ON mbt.CreditAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.isCardEnabled();

CREATE TRIGGER enabledCardBooking
    BEFORE INSERT OR UPDATE ON mbt.BookingAction
    FOR EACH ROW
    EXECUTE PROCEDURE mbt.isCardEnabled();

-- Query Test

INSERT INTO mbt.Hire (card, organization, bike, docking_point_unlock, date_unlock) VALUES
```

```
('1156896998', 3, 2, 8, '2018-05-20 08:22:54+02');
```