

## Évaluation SQL (3/2025) — Synthèse sur les Triggers

*Durée : 1h | Documents non autorisés | Répondre sur le sujet*

Nom :	Prénom :	Spécialité :	Groupe :	Note :
		<input type="checkbox"/> IA <input type="checkbox"/> RSD		/20

### Contexte : Application « QuickFood » — Livraison de repas

QuickFood est une application de livraison de repas à domicile. La base gère les clients, les livreurs et les commandes.

### Schéma de la base de données

```
CREATE TABLE Client (  
    id_client          INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    nom                VARCHAR(50) NOT NULL,  
    telephone          VARCHAR(15) UNIQUE,  
    adresse             VARCHAR(200),  
    date_inscription   DATE DEFAULT CURRENT_DATE,  
    nb_commandes        INTEGER DEFAULT 0,  
    statut              VARCHAR(15) DEFAULT 'NOUVEAU'    -- NOUVEAU, REGULIER, VIP  
);  
  
CREATE TABLE Livreur (  
    id_livreur          INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    nom                 VARCHAR(50) NOT NULL,  
    telephone           VARCHAR(15) UNIQUE,  
    disponible          BOOLEAN DEFAULT TRUE,  
    nb_livraisons        INTEGER DEFAULT 0,  
    note_moyenne         DECIMAL(3,2) DEFAULT 0    -- Note sur 5  
);  
  
CREATE TABLE Commande (  
    id_commande          INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    id_client             INTEGER REFERENCES Client(id_client),  
    id_livreur            INTEGER REFERENCES Livreur(id_livreur),  
    date_commande         TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    montant               DECIMAL(8,2) NOT NULL,  
    statut                VARCHAR(15) DEFAULT 'EN_ATTENTE',  
                        -- EN_ATTENTE, EN_COURS, LIVREE, ANNULEE  
    note_client           INTEGER CHECK (note_client BETWEEN 1 AND 5)  
);
```

**Règles métier à automatiser :**

1. Le montant minimum d'une commande est 500 DA
2. On ne peut commander qu'auprès d'un livreur disponible
3. Un client devient REGULIER après 5 commandes, VIP après 20 commandes
4. Un livreur devient indisponible quand il prend une commande
5. La note moyenne du livreur est recalculée après chaque livraison notée
6. Un client VIP ne peut pas être supprimé

**Aide-mémoire**

BEFORE	AFTER
<ul style="list-style-type: none"> <li>• Peut modifier NEW</li> <li>• RETURN NEW → valide</li> <li>• RETURN NULL → annule</li> </ul>	<ul style="list-style-type: none"> <li>• NEW en lecture seule</li> <li>• Idéal pour cascade/log</li> <li>• RETURN ignoré</li> </ul>

INSERT	UPDATE	DELETE
NEW ✓ OLD ✗	NEW ✓ OLD ✓	NEW ✗ OLD ✓

**Partie 1 : QCM (20 questions × 1 pt = 20 pts)**

Cochez la bonne réponse pour chaque question.

**1. Qu'est-ce qu'un trigger en SQL ?**

- ☐ A) Une requête SELECT automatique
- ☐ B) Un programme qui s'exécute automatiquement sur un événement
- ☐ C) Une contrainte d'intégrité
- ☐ D) Un index sur une table

**2. Quels événements peuvent déclencher un trigger ?**

- ☐ A) SELECT, INSERT, UPDATE
- ☐ B) INSERT, UPDATE, DELETE
- ☐ C) CREATE, ALTER, DROP
- ☐ D) BEGIN, COMMIT, ROLLBACK

**3. Quelle est la différence principale entre BEFORE et AFTER ?**

- ☐ A) BEFORE est plus rapide
- ☐ B) BEFORE permet de modifier NEW, AFTER non
- ☐ C) AFTER permet de modifier OLD
- ☐ D) Il n'y a aucune différence

**4. Que se passe-t-il si un trigger BEFORE INSERT retourne NULL ?**

- ☐ A) Une erreur est levée
- ☐ B) L'insertion est annulée silencieusement
- ☐ C) NULL est inséré dans toutes les colonnes
- ☐ D) Le trigger suivant s'exécute

**5. Dans un trigger sur UPDATE, quelles variables sont disponibles ?**

- ☐ A) Seulement NEW
- ☐ B) Seulement OLD
- ☐ C) OLD et NEW
- ☐ D) Ni OLD ni NEW

**6. Quel est l'ordre d'exécution des triggers ?**

- ☐ A) AFTER → Opération → BEFORE
- ☐ B) BEFORE → Opération → AFTER
- ☐ C) L'ordre est aléatoire
- ☐ D) Ils s'exécutent en parallèle

**7. Comment créer un trigger sur INSERT et UPDATE ?**

- ☐ A) C'est impossible
- ☐ B) BEFORE INSERT AND UPDATE
- ☐ C) BEFORE INSERT OR UPDATE
- ☐ D) BEFORE INSERT, UPDATE

**8. Quelle variable indique l'événement déclencheur ?**

- ☐ A) TRIGGER\_EVENT
- ☐ B) TG\_OP
- ☐ C) EVENT\_TYPE
- ☐ D) TG\_ACTION

**9. Dans un trigger AFTER DELETE, quelle est la valeur de NEW ?**

- ☐ A) La ligne supprimée
- ☐ B) NULL
- ☐ C) Une ligne vide
- ☐ D) Identique à OLD

**10. Quelle clause filtre le déclenchement d'un trigger ?**

- ☐ A) WHERE
- ☐ B) WHEN
- ☐ C) FILTER
- ☐ D) HAVING

**11. Que fait RAISE EXCEPTION dans un trigger ?**

- ☐ A) Affiche un message d'information
- ☐ B) Annule l'opération avec une erreur
- ☐ C) Passe au trigger suivant
- ☐ D) Termine la transaction

**12. Que doit retourner un trigger BEFORE DELETE pour autoriser la suppression ?**

- ☐ A) NEW
- ☐ B) OLD
- ☐ C) TRUE
- ☐ D) 1

**13. Quelle est la différence entre FOR EACH ROW et FOR EACH STATEMENT ?**

- ☐ A) ROW s'exécute une fois par ligne, STATEMENT une fois par requête
- ☐ B) STATEMENT est plus rapide
- ☐ C) ROW ne permet pas d'accéder à NEW
- ☐ D) Il n'y a pas de différence

**14. Dans un trigger BEFORE INSERT, peut-on modifier NEW.montant ?**

- ☐ A) Non, NEW est en lecture seule
- ☐ B) Oui, et la modification sera enregistrée
- ☐ C) Oui, mais la modification sera ignorée
- ☐ D) Seulement avec AFTER INSERT

**15. Comment désactiver temporairement un trigger ?**

- ☐ A) DROP TRIGGER ... TEMPORARY
- ☐ B) ALTER TRIGGER ... DISABLE
- ☐ C) ALTER TABLE ... DISABLE TRIGGER ...
- ☐ D) SET TRIGGER = OFF

**16. Que représente OLD dans un trigger UPDATE ?**

- ☐ A) La nouvelle valeur de la ligne
- ☐ B) L'ancienne valeur de la ligne avant modification
- ☐ C) La valeur par défaut
- ☐ D) La première ligne de la table

**17. Peut-on appeler une fonction depuis un trigger ?**

- ☐ A) Non, c'est interdit
- ☐ B) Oui, avec SELECT ou PERFORM
- ☐ C) Seulement les fonctions système
- ☐ D) Seulement dans AFTER

**18. Que fait RAISE NOTICE ?**

- ☐ A) Annule l'opération
- ☐ B) Affiche un message sans interrompre
- ☐ C) Enregistre dans une table de log
- ☐ D) Envoie un email

**19. Quel type doit retourner une fonction trigger ?**

- ☐ A) INTEGER
- ☐ B) BOOLEAN
- ☐ C) TRIGGER
- ☐ D) VOID

**20. Dans quel cas utilise-t-on principalement un trigger AFTER ?**

- ☐ A) Pour valider les données
- ☐ B) Pour modifier les données avant insertion
- ☐ C) Pour des actions de cascade ou journalisation
- ☐ D) Pour bloquer une opération

**Partie 2 : Compléter les Triggers (4 × 5 pts = 20 pts)****Exercice 1 — BEFORE INSERT : Valider une commande**

Ce trigger vérifie que le montant  $\geq 500$  DA et que le livreur est disponible.

```

CREATE OR REPLACE FUNCTION fn_valider_commande()
RETURNS _____ AS $$
DECLARE
    v_disponible BOOLEAN;
BEGIN
    -- Vérifier le montant minimum (500 DA)
    IF _____ < 500 THEN
        RAISE _____ 'Montant minimum : 500 DA';
    END IF;

    -- Récupérer la disponibilité du livreur
    SELECT disponible INTO v_disponible
    FROM _____
    WHERE _____ = NEW.id_livreur;

    -- Vérifier que le livreur est disponible
    IF NOT v_disponible THEN
        RAISE EXCEPTION 'Ce livreur n'est pas disponible';
    END IF;

    RETURN _____;
END;
$$ LANGUAGE _____;

CREATE TRIGGER tg_valider_commande
_____ INSERT ON _____
FOR EACH _____
EXECUTE _____ fn_valider_commande();

```

**Exercice 2 — AFTER INSERT : Mettre à jour après une commande**

Ce trigger incrémente nb\_commandes du client, met à jour son statut, et rend le livreur indisponible.

```

CREATE OR REPLACE FUNCTION fn_apres_commande()
RETURNS _____ AS $$
DECLARE
    v_nb INTEGER;
BEGIN
    -- Incrémenter nb_commandes du client
    UPDATE _____
    SET nb_commandes = nb_commandes + 1
    WHERE _____ = NEW.id_client
    RETURNING nb_commandes INTO v_nb;

    -- Mettre à jour le statut selon le nombre de commandes
    IF v_nb >= 20 THEN
        UPDATE Client SET statut = '_____'
        WHERE id_client = NEW.id_client;
    ELSIF v_nb >= 5 THEN
        UPDATE Client SET statut = '_____'
        WHERE id_client = NEW.id_client;
    END IF;

    -- Rendre le livreur indisponible
    UPDATE _____
    SET _____ = FALSE
    WHERE id_livreur = _____;

    RETURN _____;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_apres_commande
_____ INSERT ON Commande
FOR EACH ROW
EXECUTE FUNCTION fn_apres_commande();

```

**Exercice 3 — BEFORE UPDATE : Gérer la livraison**

**Quand le statut passe à 'LIVREE' : incrémenter nb\_livraisons, remettre le livreur disponible, et recalculer sa note moyenne.**

```

CREATE OR REPLACE FUNCTION fn_gerer_livraison()
RETURNS TRIGGER AS $$
DECLARE
    v_moyenne DECIMAL(3,2);
BEGIN
    -- Traiter seulement si statut passe à 'LIVREE'
    IF _____ .statut = 'LIVREE'
    AND _____ .statut <> 'LIVREE' THEN
        -- Mettre à jour le livreur
        UPDATE _____
        SET nb_livraisons = nb_livraisons + 1,
            _____ = TRUE
        WHERE id_livreur = NEW.id_livreur;

        -- Si une note est donnée, recalculer la moyenne
        IF _____ .note_client IS NOT NULL THEN
            SELECT AVG(_____) INTO v_moyenne
            FROM Commande
            WHERE id_livreur = NEW.id_livreur
            AND note_client IS NOT NULL;
            UPDATE Livreur
            SET _____ = v_moyenne
            WHERE id_livreur = NEW.id_livreur;
        END IF;
    END IF;
    RETURN _____;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_gerer_livraison
_____ UPDATE ON _____
FOR EACH _____
EXECUTE FUNCTION fn_gerer_livraison();

```

**Exercice 4 — BEFORE DELETE : Protéger les clients VIP**

Ce trigger empêche la suppression d'un client ayant le statut 'VIP'.

```

CREATE OR REPLACE FUNCTION fn_proteger_vip()
RETURNS _____ AS $$
BEGIN
    -- Vérifier si le client est VIP
    IF _____.statut = '_____' THEN
        RAISE _____ 'Impossible de supprimer un client VIP';
    END IF;

    -- Autoriser la suppression
    RETURN _____;
END;
$$ LANGUAGE _____;

CREATE TRIGGER tg_proteger_vip
_____ DELETE ON _____
FOR EACH _____
EXECUTE _____ fn_proteger_vip();

```

***Bon courage!***