Réponse Partiel 2011-2012 en BDD

(Pour ceux qui en ont marre d'attendre un corrigé qui ne viendra jamais sur AREL)

Question 1 :

Les contraintes de clés primaires ou de clés étrangères sont des triggers particulier car en tant que **contraintes d'intégrités**, on établit une clause permettant de contraindre la modification de tables via les requêtes qu'un utilisateur fera afin que les données saisies dans la base soient conformes aux données attendues. Le trigger s'occupe lui de vérifier avant ou après qu'une ligne a été ajoutée, supprimée ou modifiée.

En gros, les contraintes vérifient bien AVANT l'ajout qu'il s'agit du format demandé.

Ps : On utilise en général les contraintes sur les clés primaires/secondaires car ça permet de ne pas se taper des clés carrément connes si on laisse le moteur de base s'en charger. Et puis ça définit des règles relatives aux valeurs autorisées dans les colonnes, c'est un mécanisme standard pour la mise en application de l'intégrité d'une BDD.

Question 2 :

Réponse en anglais

First, I'll start my answer by defining **trigger**: a **trigger** is a stored procedure that is run when a row is added, modified or deleted.

Triggers can run ***BEFORE*** the action is taken or *AFTER* the action is taken.
`BEFORE` triggers are usually used when validation needs to take place before accepting the change. They run before any change is made to the database. Let's say you run a database for a bank. You have a table `accounts` and a table `transactions`. If a user makes a withdrawal from his account, you would want to make sure that the user has enough credits in his account for his withdrawal. The `BEFORE` trigger will allow to do that and prevent the row from being inserted in `transactions` if the balance in `accounts` is not enough.

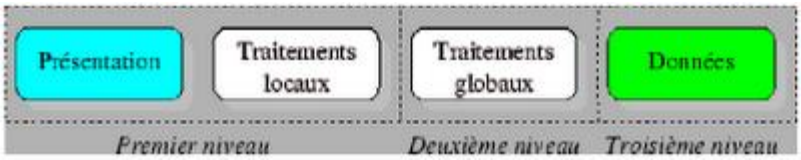
AFTER triggers are usually used when information needs to be updated in a separate table due to a change. They run after changes have been made to the database (not necessarily committed). Let's go back to our back example. After a successful transaction, you would want `balance` to be updated in the `accounts` table. An `AFTER` trigger will allow you to do exactly that.

Question 3 :

Rappel sur l'intérêt d'une architecture 3 tiers :

Dans ce but, l'architecture trois tiers applique les principes suivants :

- Les données sont toujours gérées de façon centralisée.
- La présentation est toujours prise en charge par le poste client.
- La logique applicative est prise en charge par un serveur intermédiaire.



- Premier niveau : l'affichage et les traitements locaux (contrôles de saisie, mise en forme de données…) sont pris en charge par le poste **client**.
- Deuxième niveau : les traitements applicatifs globaux sont pris en charge par le **service applicatif**.
- Troisième niveau : les services de base de données sont pris en charge par un **SGBD**.

Réponse

Le but, c'est qu'en tiers 3, on allège à mort le client (on dit qu'il est *Thin*). Donc ça veut dire que si vous déplacez un traitement du serveur WEB vers le serveur BDD, alors c'est souvent parce que vous allez réaliser un triage dans la base directement (select [blablabla] from…). Si vous avez beaucoup de données à récupérer d'un coup pour appliquer un trigger dessus ou une procédure, c'est largement plus rapide de le faire via le serveur BDD puisque le serveur WEB va devoir tout télécharger avant de faire votre requête.

Question 4 :

Définition d'un tuple :

> En mathématiques, si n est un entier naturel alors un n-uplet est une collection ordonnée de n objets, appelés composantes du n-uplet. On dit aussi n-uple. (*Wikipedia)*

En somme, ça veut dire qu'un **tuple**, c'est comme une instance. Ne pas confondre avec un **uple** qui représente lui un couple de valeur (une ligne d'une table SQL en gros). Un **tuple**, c'est la représentation de tous les couples d'une entité (donc toutes les lignes d'une table SQL).

Exemple à la con :

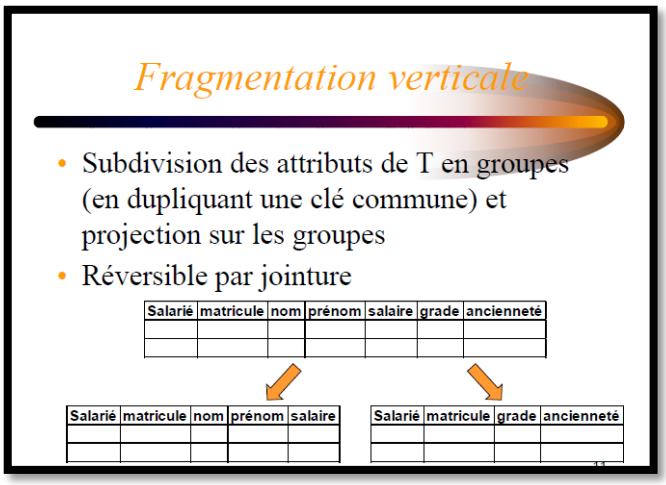
```
CREATE TABLE Etudiant (
numetud NUMBER(5) CONSTRAINT PKEtudiant PRIMARY KEY,
nom VARCHAR2(30),
prenom VARCHAR2(30),
age NUMBER(2));
```

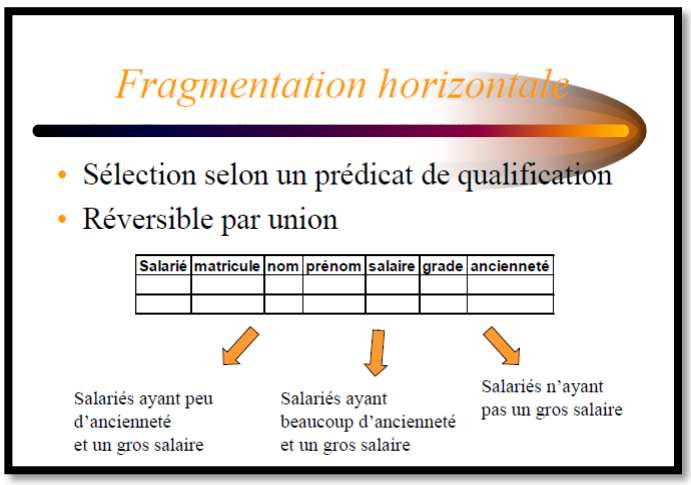
Les **uples** sont : numetud, nom, prenom et age. D'ailleurs, cette table c'est un 5-uple !
Le **tuple**, c'est une ligne de cette table.

Réponse :

- Dans une fragmentation des données en horizontale, les tuples sont réparties.
- Dans une fragmentation des données en verticale, les tuples sont découpés, cela nécessite d'avoir des colonnes communes dupliquées.

Voici deux images parfaites que j'ai trouvé sur internet pour expliquer la différence :





Question 5 :

Ici, on veut vous faire réfléchir sur l'aspect « sécurité » de votre architecture. La politique de sécurité est assez importante puisqu'elle garantie les droits d'accès aux données et aux ressources d'un système en mettant en place des mécanismes d'authentification qui permettent d'assurer que les utilisateurs possèdent uniquement des droits qui leur ont été donnés.

Réponse :

Je n'ai pas eu le temps de la rédiger, mais voici ce que j'ai trouvé : http://cyberzoide.developpez.com/securite/privileges-base-de-donnees/#LB

Question 6 :

On a plein d'autre type de SGBD ! Par exemple on a des SGBD fédérées. Elles donnent aux utilisateurs une vue unique des données implémentées sur plusieurs système à priori hétérogènes. Un cas type d'utilisation de SGBD fédérées lors de la concentration d'entreprises : faire cohabiter les différents systèmes tout en leur permettant d'interopérer quand même.

ENJOY.