

## Exceptions - Triggers

CUSTOMER\_CST => Données client

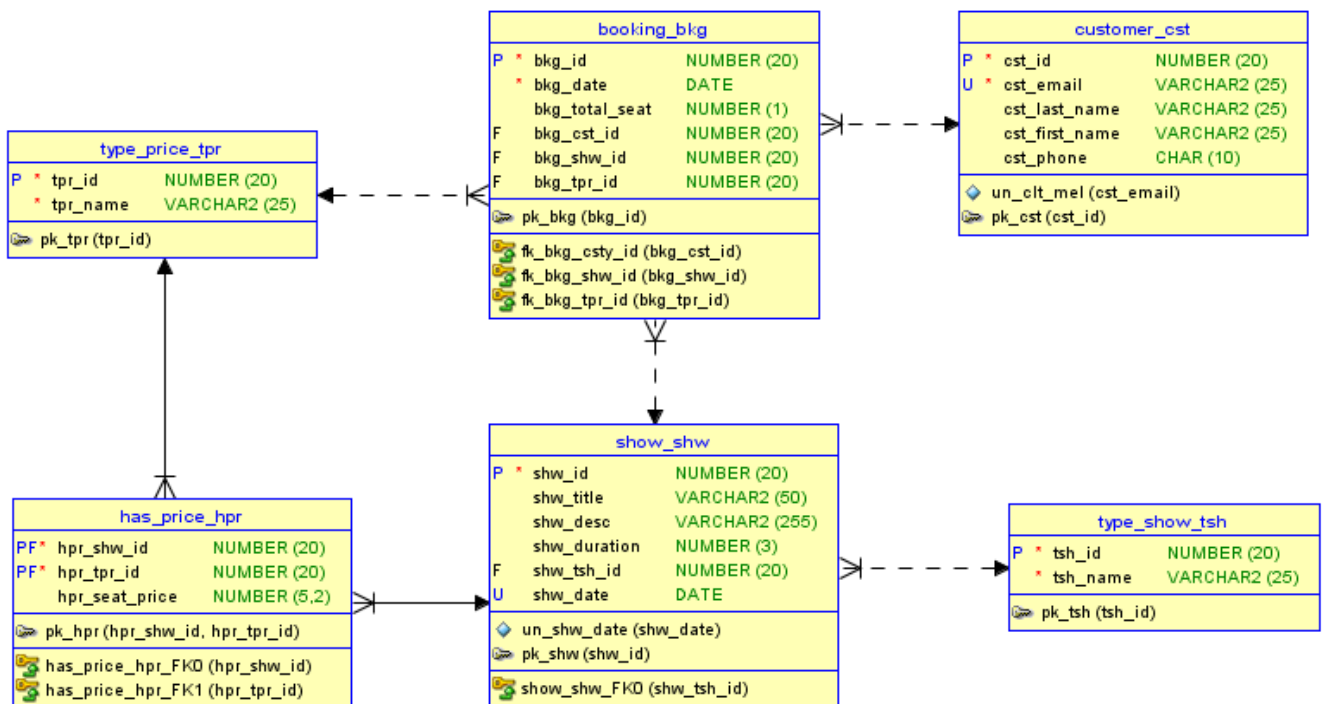
HAS\_PRICE\_HPR => Le prix d'une place de spectacle dépend du spectacle et du type de tarif

SHOW\_SHW => Données spectacle

TYPE\_PRICE\_TPR => Type de tarif

TYPE\_SHOW\_TSH => Type de spectacle

BOOKING\_BKG => Données de réservation. bkg\_total\_seat est le nombre de places réservées pour un client et un spectacle, avec un type de tarif



### 1. Curseur explicite

Créer un bloc anonyme (avec une section exception) qui affiche le nombre de spectacles par type de spectacle.

### 2. Procédure – Erreur oracle – Erreur utilisateur

Ecrire une procédure qui ajoute un nouveau client (la valeur de clé primaire est générée automatiquement).

Elle gèrera les cas d'erreurs suivants (afficher un message de votre choix) :

- Erreur générée par Oracle lors d'une violation de la contrainte UNIQUE
- Erreur générée par Oracle lors d'une violation de la contrainte Not NULL
- A la fois nom et prénom non renseignés.

### 3. Curseur Implicite – raise\_application\_error

Créer une procédure qui modifie les prix d'un spectacle (connu par son identifiant), selon un certain coefficient (également passé en paramètre).

Le nombre des lignes mises à jour sera affiché.

La section exception gérera le cas où aucune ligne n'est modifiée en affichant un message aussi explicite que possible

#### 4. Trigger - Contrôler des modifications

Créer un trigger qui empêche toute modification de prix sur des spectacles qui se sont déjà déroulés. Le trigger utilisera `raise_application_error`.

#### 5. Tracer des modifications

Créer la table dans laquelle les changements sur `booking_bkg` (UPDATE, DELETE uniquement) seront tracées. Elle comportera au moins : utilisateur ayant modifié la table, date de modification, évènement ayant généré le déclenchement du trigger ( soit UPDATE, soit DELETE), identifiant de la réservation concernée. Par exemple :

```
CREATE TABLE log_bkg_lgb (
    lgb_date DATE,
    lgb_user VARCHAR2(30),
    lgb_event VARCHAR2(30),
    lgb_bkg_id NUMBER(20)
);
```

Ecrire un trigger qui insère une ligne dans cette table à chaque fois qu'une suppression ou une modification (update) a lieu sur la table des réservations.

Ecrire un bloc anonyme qui contiendra la ou les instructions SQL permettant de déclencher le trigger.

Vérifier la mise à jour des tables concernées.

#### 6. Trigger – calculer des valeurs automatiquement

Modifier la table des réservations : ajouter une colonne pour stocker le prix total d'une réservation.

Exemple :

```
ALTER TABLE booking_bkg ADD bkg_total_amount NUMBER(6,2);
```

Ecrire un trigger qui calcule le montant total d'une réservation à chaque fois qu'une réservation est insérée ou modifiée (modification des colonnes `bkg_total_seat`, `bkg_tpr_id` et `bkg_shw_id` seulement). Si besoin vous pouvez utiliser `IF UPDATING('NOM_COLONNE')` THEN pour tester quelle colonne a été mise à jour.

Votre trigger comportera une section exception avec l'appel de `RAISE_APPLICATION_ERROR` lorsque le prix total ne peut pas être calculé (par exemple `SELECT INTO` ne retourne aucune ligne).

#### 7. Curseurs multiples avec paramètres – GROUP BY

Créer un bloc anonyme qui calcule la moyenne des durées de spectacle par type de spectacle, et affiche, pour chaque type de spectacle, la liste de ceux qui ont une durée supérieure à la moyenne

La résultat devra être semblable à :

THÉÂTRE 85	=> type of show – average duration
Show title : CALACAS - ZINGARO 90	=> show title of type Theatre – duration > computed avg
Show title : LES AVEUGLES 90	=> show title of type Theatre – duration > computed avg

Et ainsi de suite.