IFT2935 - Bases de données

Professeur: François Major

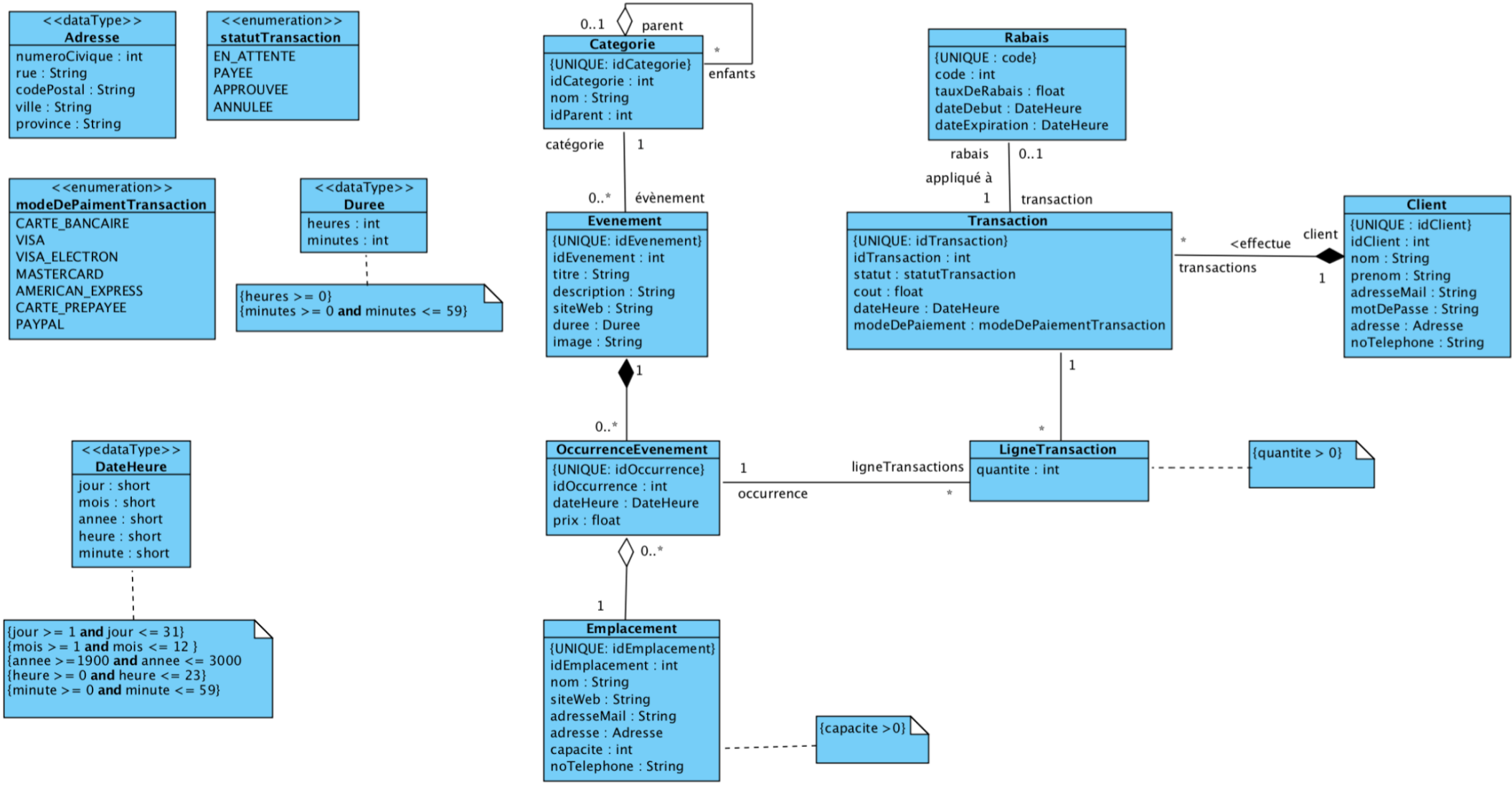
Travail Pratique #2

6 décembre 2016

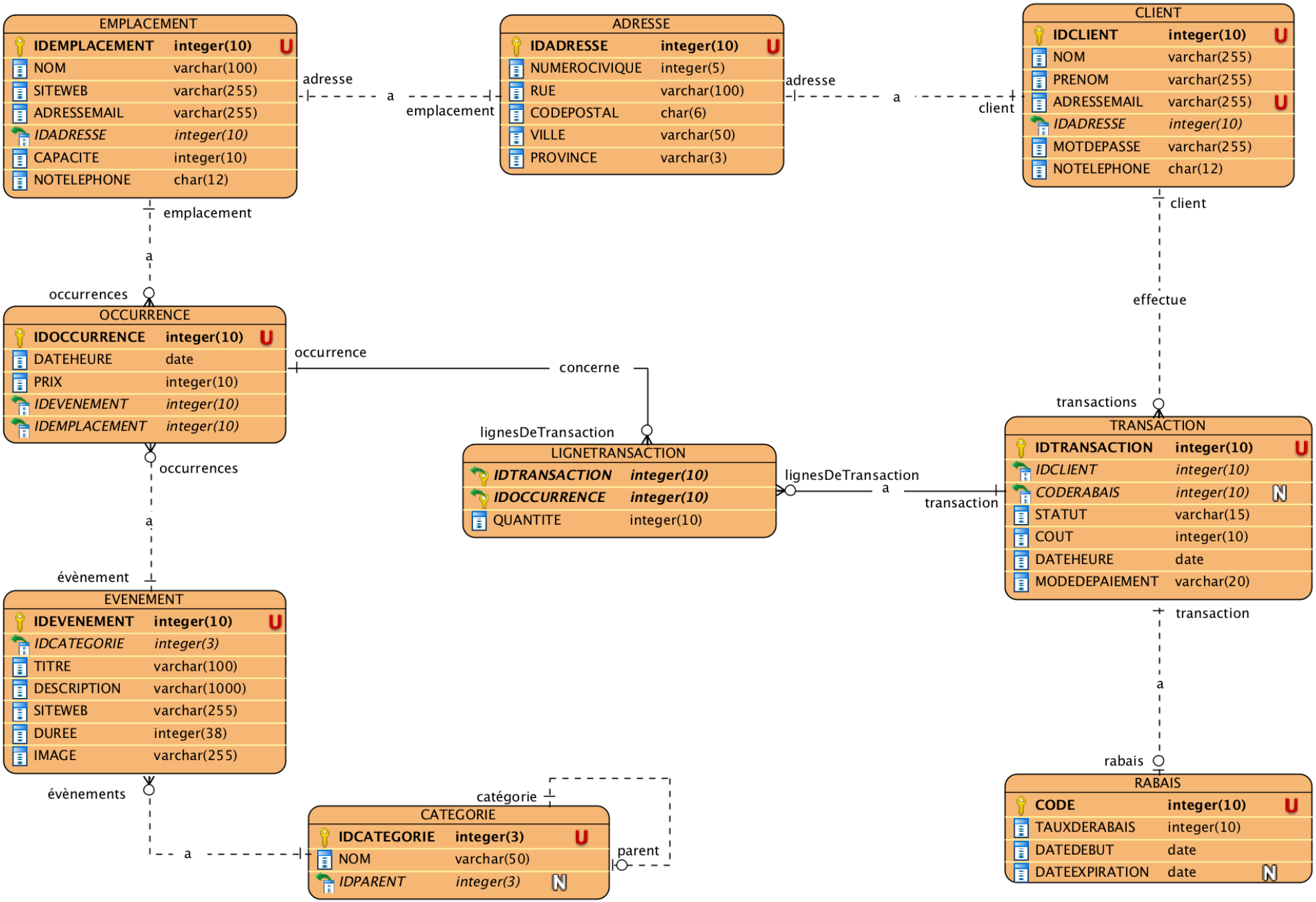
Nom: Paul Chaffanet
Matricule: 1009543

Nom: Samuel Guigui
Matricule: 20035458

Diagramme de classe `e-ticket`



Modèle entité-association de la base de donnée relationelle `e-ticket`



2) Spécifier les différentes contraintes d'intégrité.

CATEGORIE

`cs_doublonCat`
On vérifie qu'il n'existe pas deux catégories qui ont exactement le même nom et le même parent. On évite ainsi les catégories doublons (on limite les erreurs).

ADRESSE

`cs_doublon`
On met une contrainte UNIQUE sur toutes les colonnes de la table sauf sur IDADRESSE. On évite ainsi l'insertion d'adresse doublon, et donc on évite de surcharger la base de donnée.

`cs_numeroCivique`
On met une contrainte pour vérifier que le numeroCivique a une valeur supérieure à 0.

`cs_codePostal`
Contrainte pour vérifier que le code postal respecte le format canadien (e.g 'X1X1X1').

`cs_ville`
Contrainte pour vérifier qu'une ville respecte un certain format. La chaîne doit commencer par une majuscule. Elle peut être de forme 'Montréal' ou 'Saint Jean' ou 'Saint-Jean' ou 'Saint-Jean-sur-Richelieu' par exemple.

`cs_province`
Contrainte pour vérifier que le sigle provincial et bien une province canadienne (e.g. 'QC', 'ON', etc.).

EVENEMENT

`cs_doublonEvenement`
On interdit l'insertion d'évènement identique en tout point (créations d'évènements doublons).

`cs_dureePositive`
Un évènement doit avoir une durée positive.

EMPLACEMENT

`cs_adressemailEmplacement`
On contraint à ce que l'adresse mail respecte le bon format (p.e 'paul.jp.chaffanet@gmail.com').

`cs_capacitePostive`
On contraint à ce que la capacité soit toujours positive pour un emplacement.

`cs_noTelephoneEmplacement`
On contraint à ce qu'un numéro de téléphone soit au format '999-999-9999' par exemple.

Trigger

`AUemplacement`
Si on met à jour la capacité d'un emplacement, on vérifie que la nouvelle capacité est assez grande pour contenir tous les billets déjà vendus (c-à-d les lignesTransaction associées à des transactions 'payée') pour toutes les occurrences associées à ces emplacements.
Si la nouvelle capacité n'est pas assez grande, on refuse la mise à jour de capacité et on signale à l'utilisateur l'erreur générée.

OCCURRENCE

`cs_prixPositif`
Le prix d'un évènememement peut être 0 ou plus. En effet, il existe bien des évènements gratuits pour lesquels des billets pourraient être émis. En revanche, les prix négatifs sont bien sûrs incohérents.

Trigger

`BUdateHeureOccurrence`
À la mise à jour de la dateHeure d'une occurrence, on contraint à ce que chaque ligne transaction qui concerne cette occurrence aient une DATEHEUREligneTransaction < DATEHEUREOccurrence.
Sinon il y aurait incohérence (transaction effectuée après le début d'une occurrence est impossible!). Après qu'une occurrence a débuté, la vente de billets est close, il ne peut donc exister de transactions concernant cette occurrence.

`AUemplacementOccurrence`
Après la mise à jour d'un emplacement pour une occurrence, on vérifie que la nouvelle capacité est assez grande pour contenir tous les billets déjà vendus (c-à-d les lignesTransactions associées à des transactions 'payées' et associée à cette occurrence).
Sinon la capacité est insuffisante pour le nouvel emplacement et on interdit la mise à jour de l'emplacement.

`AUprixOccurrence`
Si on met à jour le prix d'occurrences, on update le coût des transactions 'en attente' associées à ces occurrences.
Les transactions 'payées', 'annulées' et 'approuvées' ont été effectuées à un prix convenu, on ne modifie donc pas leur coût.

CLIENT

`cs_uniqueMail`
Le mail est la seule donnée qui doit être unique pour qu'un client puisse être inséré. En effet, on ne peut pas créer de comptes clients qui partagent le même identifiant de connexion.

`cs_nomClient`
Un NOM est une chaîne de caractères qui ne contient pas de nombre et qui commence par une lettre majuscule.

`cs_prenomClient`
Un PRENOM est une chaîne de caractères qui ne contient pas de nombre et qui commence par une lettre majuscule.

`cs_adresseMailClient`
On contraint à ce que l'adresse mail respecte le bon format (p.e 'paul.jp.chaffanet@gmail.com').

`cs_noTelephoneClient`
On contraint à ce qu'un numéro de téléphone soit au format '999-999-9999'.

RABAIS

`cs_tauxDeRabais`
On contraint à ce que le taux de rabais soit compris entre 1% à 99%.

`cs_datesRabais`
On vérifie que la dateDebut du rabais soit inférieure à la dateExpiration d'un rabais.

Trigger

`BIUdatesRabais`
S'il existe une transaction (quelque soit son statut) utilisant ce rabais telle que dateDebutRabais > dateTransaction ou dateTransaction > dateExpirationRabais, alors on interdit la mise à jour des dates du rabais.

`BUtauxRabais`
S'il existe des transactions avec un statut 'payée', 'approuvée' ou 'annulée' utilisant le rabais pour lequel on veut mettre à jour le taux, on interdit la mise à jour du taux de rabais.
Si ce rabais n'est utilisé que par des transactions avec un statut 'en attente', on autorise la mise à jour du taux de rabais.

`AUtauxRabais`
On met à jour le coût des transactions 'en attente' avec le nouveau taux de rabais.

TRANSACTION

`cs_statut`
Une transaction ne peut avoir que 4 statuts (`'annulée'`, `'en attente'`, `'approuvée'`, `'payée'`)

`cs_modeDePaiement`
Le mode de paiement est contraint à seulement quelques modes de paiements autorisés, à savoir, par exemple `'PAYPAL'` ou `'VISA'`.

Trigger

`BUrabaisTransaction`
On interdit modifier le rabais d'une transaction dont le statut est 'payée', 'approuvée' ou 'annulée'. La transaction est passée, elle ne peut donc plus être modifiée.

`AUrabaisTransaction`
Mise à jour du coût si changement du coderabais utilisé lors d'une transaction 'en attente'.

`BItransaction`
Une transaction doit avoir un statut 'en attente' à l'insertion. En effet, la transaction n'est liée à aucune ligne transaction, elle ne peut donc avoir un statut 'payée', 'annulée' ou 'approuvée'.

`BUtransaction`
Avant de mettre à jour une transaction, si celle-ci était 'payée' ou 'annulée' alors on ne peut plus la modifier.
Si la transaction est 'approuvée', alors on peut changer son statut à 'annulée' ou 'payée' mais la transaction ne peut pas retourner au statut 'en attente'.
On contraint également dans ce trigger à ne pas pouvoir changer le coût, le mode de paiement, ou l'idClient d'une transaction dont le statut est 'approuvée'. En effet, nous considérons qu'une transaction 'approuvée' est une transaction dont on attend juste la réception des fonds pour que la réservation/la vente des billets soit effective.

`AIUstatutTransaction`
Après la mise à jour de statut de transactions, on vérifie que les transactions qui ont un statut 'payée' ne font pas dépasser la capacité des occurrences auxquelles les transactions sont reliées. Sinon on ne peut pas changer le statut de la transaction à 'payée'. En effet, ce n'est qu'à partir du moment où une transaction est 'payée' que l'on considère la vente du billet comme effective. C'est donc au moment de ce changement de statut que l'on vérifie qu'il reste assez de places disponibles pour toutes les occurrences auxquelle le client a acheté des places.
De plus, on vérifie que des lignes transactions existent pour les transactions qui ont un statut 'payée', 'approuvée', ou 'annulée'. Sinon on interdit également la mise à jour du statut (La transaction doit rester 'en attente'.

`BUdateHeureTransaction`
On ne peut pas modifier la dateHeure d'une transaction avec un statut 'payée' ou 'annulée'
On autorise la modification de la dateHeure d'une transaction 'approuvée' dans la mesure où elle peut désigner l'heure d'un changement de statut vers 'payée' ou 'annulée'.
La nouvelle dateHeure ne peut pas être inférieure à l'ancienne dateHeure quelque soit le statut de la transaction.
La dateheure de la transaction doit être, de plus, inférieures à tous les dateheures des occurrences (on regarde les lignestransactions associées à cette transaction pour trouver ces occurrences) . En effet, l'occurrence ne doit pas être passée pour pouvoir acheter des tickets.

LIGNETRANSACTION

`cs_quantitePositive`
Une ligne de transaction doit avoir forcément une quantité strictement positive pour pouvoir être insérée.

Trigger

`BIUligneTransaction`
On vérifie que la clé étrangère existe bien pour IDOCCURRENCE et IDTRANSACTION.
On vérifie avant d'insérer une ligne transaction, ou d'update son idoccurrence, que la dateHeureOccurrence > dateHeureTransaction.

`BDligneTransaction`
On ne peut pas supprimer des billets pour une transaction 'annulée', 'payée' ou 'approuvée'. Les lignes d'une transaction 'payée', 'annulée' et 'approuvée' ne sont pas modifiables afin de limiter le risque d'erreurs.

`AIUDligneTransaction`
On met à jour le coût des transaction 'en attente' en fonction de la quantite demandée et du prix de l'occurence.
On ne modifie pas le coût des transactions 'payée', 'en attente', ou 'approuvée' car ce sont des transactions passées.
On ne fait pas de vérification de capacité. Ce n'est qu'à partir du moment que la transaction est 'payée' que l'on vérifie si la capacité de l'emplacement est suffisante pour délivrer les billets. Sinon la transaction ne pourra pas passer au statut 'payée'.

Modèle relationnel de la base de données `e-Ticket`