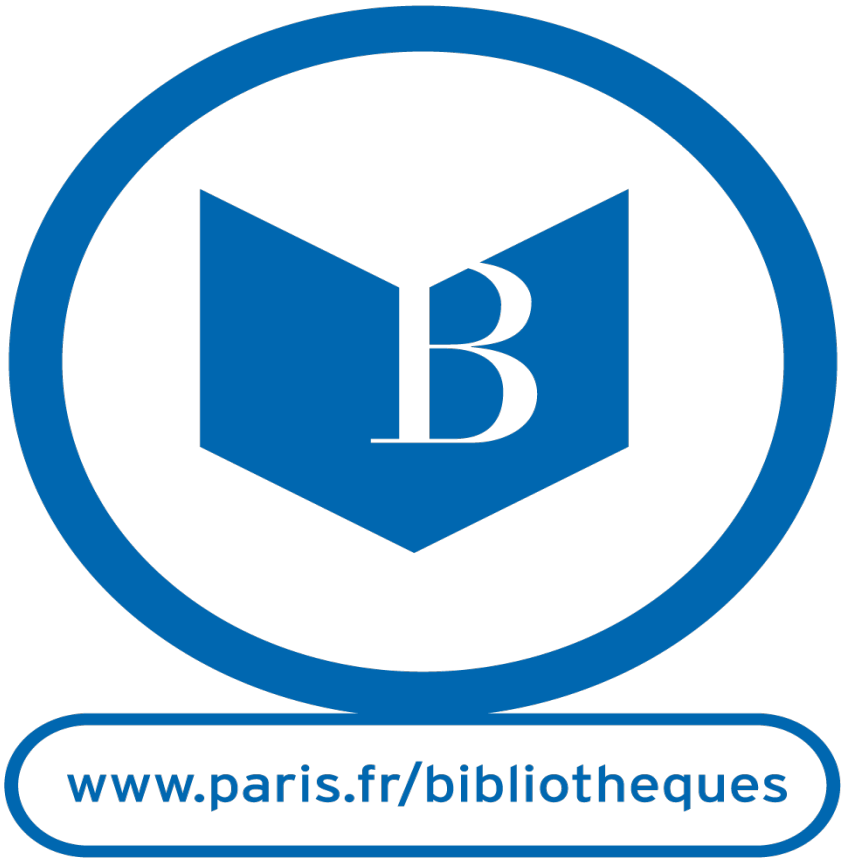
Rapport

Base De Données

Bibliothèques

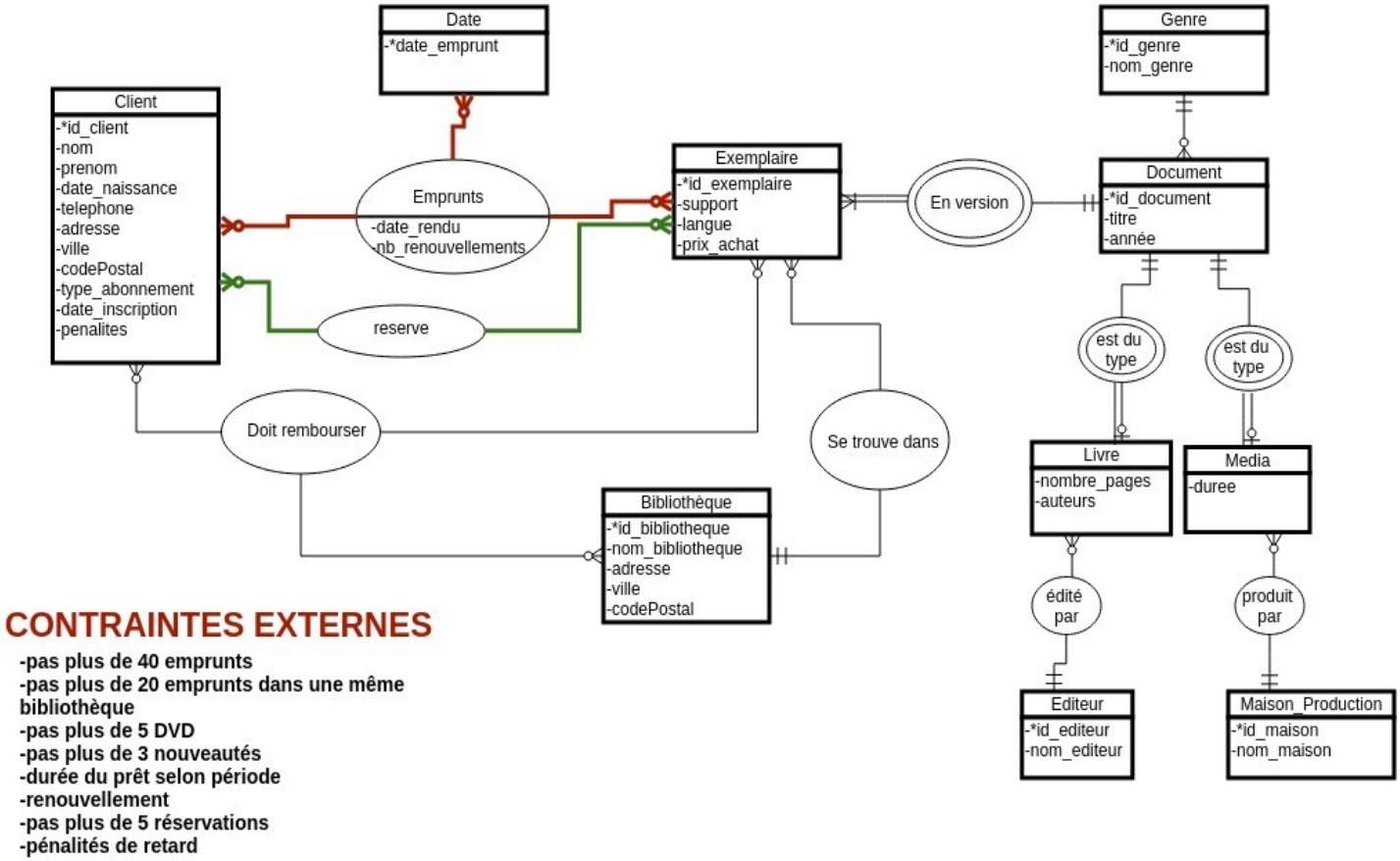


ELKRIEFF Benjamin et MUSENGA Christ

M1 INFORMATIQUE

SOMMAIRE

1. Simplifications choisies



Dans ce projet, plusieurs simplifications par rapport à notre modélisation a été faite. En effet nous n'avons pas forcément utilisé toutes les tables ou tous les attributs de chaque table pour simuler les actions des bibliothèques par l'utilisation des triggers.

Nous avons en majorité utilisé les tables :

- ***client***
- ***bibliotheque***
- ***exemplaire***
- ***emprunts***

- ***reserve***
- ***date_courante***

De plus, pour les emprunts en été nous avons plutôt préféré faire en sorte que les emprunts en été soient faites entre le mois de Juin et le mois de Septembre plutôt qu'ils soient faite plus précisément entre le 21 Juin et le 20 Septembre.

Aussi, nous avons fait en sorte que les clients ne peuvent pas emprunter ou réserver pendant la période des 3 semaines précédent la fin de leur inscription car ils doivent d'abord renouveler leur abonnement (sinon certains pourraient profiter de faire exprès d'emprunter la veille de leur fin d'inscription).

De plus, pour savoir si un exemplaire n'est pas encore rendu, nous avons préféré mettre la colonne ***date_rendu*** à **NULL**. Si cet exemplaire est rendu alors ***date_rendu*** vaudra la date où on veut rendre l'exemplaire.

Aussi, nous avons préféré faire en sorte qu'un exemplaire est nouveau si la différence entre sa différence entre sa date d'entrée dans la bibliothèque et la date courante est de **7 jours**.

Nous avons aussi fait en sorte que lorsqu'un client **A** réserve un exemplaire **X** emprunté, au moment où cet exemplaire **X** sera rendu, cet exemplaire sera automatiquement emprunté par le client **A**.

Enfin, si un client est arrivé à terme de son abonnement mais qu'il n'a pas rendu des exemplaires en retard, lorsque que celui-ci aura rendu ses exemplaires, il faudra attendre le lendemain pour qu'il soit supprimé de la base de de données.

2. Listes des fonctions

Pour ce projet, nous avons implémenté plusieurs fonctions afin que la majorité soit utilisé dans les triggers :

- **nb_emprunts_biblio(idBiblio INTEGER)** : prend en argument l'id d'une bibliothèque et retourne son nombre d'emprunts dans celle-ci.
- **nb_emprunts_total()** : retourne le nombre d'emprunt total dans toutes bibliothèques.
- **abonnement_ok(idClient INTEGER, idExemplaire INTEGER)** : prend en argument l'id d'un client et l'id d'un exemplaire puis vérifie si ce client peut emprunter cet exemplaire par rapport à son abonnement(normale,CD ou DVD).
- **get_duree_emprunt(d DATE)** : prend en argument une date(date d'emprunt plus précisément) et renvoie son nombre de jours d'emprunt selon la période de l'année(durée d'emprunt différente selon la période).
- **is_late_exemplaire(idClient INTEGER, idExemplaire INTEGER)** : prend en argument l'id d'un client et l'id d'un exemplaire et vérifie si cet exemplaire du client est en retard.
- **is_late(idClient INTEGER)** : prend en argument l'id d'un client et vérifie si ce client est en retard pour ses rendus d'emprunts.
- **nb_jours_en_retard(idExemplaire INTEGER)** : prend en argument l'id d'un exemplaire et retourne le nombre de jours de retard de cet exemplaire(si il est en retard bien évidemment) depuis la date prévu de retour de cet exemplaire.

- **est_a_3semaines_finAbo(idClient INTEGER)** : prend en argument l'id d'un client et vérifie si ce client est dans la période des 3 semaines précédent la fin de son inscription.
- **max_limite_emprunts_40(idClient INTEGER)** : prend en argument l'id d'un client et vérifie si ce client peut encore emprunter par rapport au nombre d'emprunts max (40 en tout).
- **max_limite_emprunts_biblio(idClient INTEGER, idBiblio INTEGER)** : prend en argument l'id d'un client et l'id d'une bibliothèque et vérifie si ce client peut encore emprunter dans cette même bibliothèque par rapport au nombre d'emprunts max (20 en tout).
- **max_limite_emprunts_DVD(idClient INTEGER)** : prend en argument l'id d'un client et vérifie si ce client peut encore emprunter un nouveau DVD par rapport au nombre max d'emprunts de DVD (5 en tout).
- **max_limite_emprunts_nouveaute(idClient INTEGER)** : prend en argument l'id d'un client et vérifie si ce client peut encore emprunter une nouveauté par rapport au nombre max d'emprunts de nouveautés (3 en tout).
- **max_limite_emprunts_ok(idClient INTEGER, idExemplaire INTEGER, idBibliotheque INTEGER)** : vérifie si un client donné respecte bien toutes les différentes limites d'emprunt (4 fonctions précédentes ci-dessus).
- **max_limite_reservation(idClient INTEGER)** : prend en argument l'id d'un client puis vérifie si ce client peut encore emprunter par rapport au nombre de réservations max (5 en tout).

- **actualisationPenalites(idClient INTEGER)** : prend en argument l'id d'un client et fait la mise à jour des pénalités de ce client par rapport aux exemplaires en retard du client et au nombre de jours de retard pour chacun des exemplaires (grâce à la fonction *nb_jours_en_retard()*).

- **actualisationPenalitesDefaut()** : mise a jour des pénalités de tous les clients en retard, par défaut en utilisant la fonction *actualisationPenalites()* ci-dessus.

- **alerteRendreDans2jours(idClient INTEGER)** : prend en argument l'id d'un client et va avertir que ce client doit rendre le livre dans 2 jours.
- **penalites_accumulees(idClient INTEGER, n INTEGER, prev DATE)** : prend en argument l'id d'un client, un nombre de jours **n** et une date **prev** puis retourne les pénalités accumulées de ce client, en fonction de ce nombre de jours de retard rajouté **n**. L'argument **prev** sert ici à savoir si on a avancé la date courante à partir d'un jour avant la date butoir d'un exemplaire, d'où les pénalités seront différentes.
- **fin_abonnement(idClient INTEGER)** : prend en argument l'id d'un client et vérifie si ce client est en fin d'abonnement.
- **renouvellement_inscription(idClient INTEGER)** : prend prend en argument l'id d'un client et renouvelle son abonnement si celui-ci n'est pas en retard sur ses exemplaires et qu'il est dans la période des 3 semaines précédent la fin de son inscription (grâce à la fonction *est_a_3semaines_finAbo()*). Si l'abonnement d'un client

s'est terminé, qu'il a fini par rendre tous ses exemplaires en retard un jour **J**, il pourra (si il veut) renouveler son abonnement, sinon il sera supprimé de la table ***client*** le lendemain du jour **J**.

- **avancer_date(nb_jours INTEGER)** : prend en argument un nombre de jours et fait avancer la date courante de **nb_jours** jours.
- **reculer_date(nb_jours INTEGER)** : prend en argument un nombre de jours et fait reculer la date courante de **nb_jours** jours.

- **renouvellement_exemplaire(idExemplaire INTEGER)** : prend en argument un exemplaire et renouvelle son emprunt en updatant la date d'emprunt à la date courante si cet exemplaire n'est pas en retard et n'est pas réservé.

- **a_fait_reservation(idClient INTEGER, idExemplaire INTEGER)** : prend en argument l'id d'un client et l'id d'un exemplaire puis va vérifier si ce client donné a fait une réservation sur cet exemplaire.

3. Liste des triggers

Pour ce projet, nous avons utilisé différents triggers afin de gérer différentes actions dans la bibliothèque (emprunts, réservations etc..) :

- **emprunte()**
- **reservation()**
- **updateDateCourante()**
- **alerte_rendu_exemplaire()**

4. Règles de gestion qui sont gérés par les triggers

- **emprunte()** : Lors de l'emprunt d'un client(insertion dans la table ***emprunts***), vérifie si ce client peut emprunter un exemplaire d'un document selon les différentes conditions (pas plus de 15€ de pénalité, exemplaire pas emprunté, pas réservé etc..). Elle vérifie d'abord si le client n'est pas bloqué, si le client peut emprunter l'exemplaire par rapport à son abonnement grâce à la fonction ***abonnement_ok()***, si il n'a pas de retard grâce à la fonction ***is_late()***, si l'exemplaire n'est pas déjà emprunté ou réservé et si ce client respecte bien toutes les limites d'emprunts grâce à la fonction ***max_limite_emprunts_ok()***.
- **reservation()** : Lors de la réservation d'un client (insertion dans la table ***reserve***), elle vérifie d'abord si le client n'est pas bloqué, si le client peut emprunter l'exemplaire par rapport à son abonnement grâce à la fonction ***abonnement_ok()***, si le livre est déjà emprunté, si le client n'a pas de retard grâce à la fonction ***is_late()*** et si le client respecte bien les limites du nombre de réservations grâce à la fonction ***max_limite_reservation()***.
- **updateDateCourante()** : Lors de la mise à jour de la date courante (update dans la table ***date_courante***), il y a une mise à jour des pénalités des clients si ils sont en retard en le vérifiant grâce à la fonction ***is_late()***. Les pénalités sont modifiées grâce à la fonction ***penalites_accumulees()***. De plus, elle va aussi vérifier pour chaque client si son abonnement prend fin. Dans ce cas, si il n'a aucun exemplaire à rendre, on le supprime de la table ***client***, sinon on le garde tant qu'il n'a pas rendu ses exemplaires. Lorsque celui-ci les aura rendu, il sera supprimé de la table ***client*** le

lendemain du jour où il ses aura tous rendu. Par exemple, un client finit par rendre tous ses exemplaires non rendus le **01/07/2016**, il sera finalement supprimé de la table ***client*** le **02/07/2016**.
Enfin pour chaque client, il va vérifier si ses exemplaires empruntés sont à rendre dans 2 jours grâce à la fonction ***alerteRendreDans2jours()***.

- **alerte_rendu_exemplaire()** : Lors d'un rendu d'un exemplaire(update de la colonne ***date_rendu*** de la table ***emprunts***), va lancer une alerte comme quoi un client vient de rendre un exemplaire et si un client avait fait une réservation sur l'exemplaire rendu, ce client l'empruntera directement(insertion dans la table ***emprunts***).