BIAU-GUILHEMBET Marie
ROUSSEL Gildas
INFO 1

BASES DE DONNÉES RELATIONNELLES
COMPTE RENDU DE PROJET
MODÉLISATION DE LOCATIONS DANS UNE STATION DE SKI

Mai 2016

Table des matières

Liste des figures

On cherche à modéliser une base de données pour le compte d'une agence immobilière qui propose la location d'appartements dans la station de sports d'hiver de Valthoriaz. L'agence dispose de plusieurs appartements de différentes catégories et réparties sur plusieurs sites de la station. Après la mise en place du modèle, on le modélisera en MySQL à l'aide du logiciel MySQL Workbench. Puis on simulera l'interaction entre un client et la base de données.

1 Le modèle

Le modèle retenu est un modèle reliant cinq entités (client, appartement, site, categorie et periode) à l'aide de quatre associations (reserver, etre, situer et couter) (figure 1).

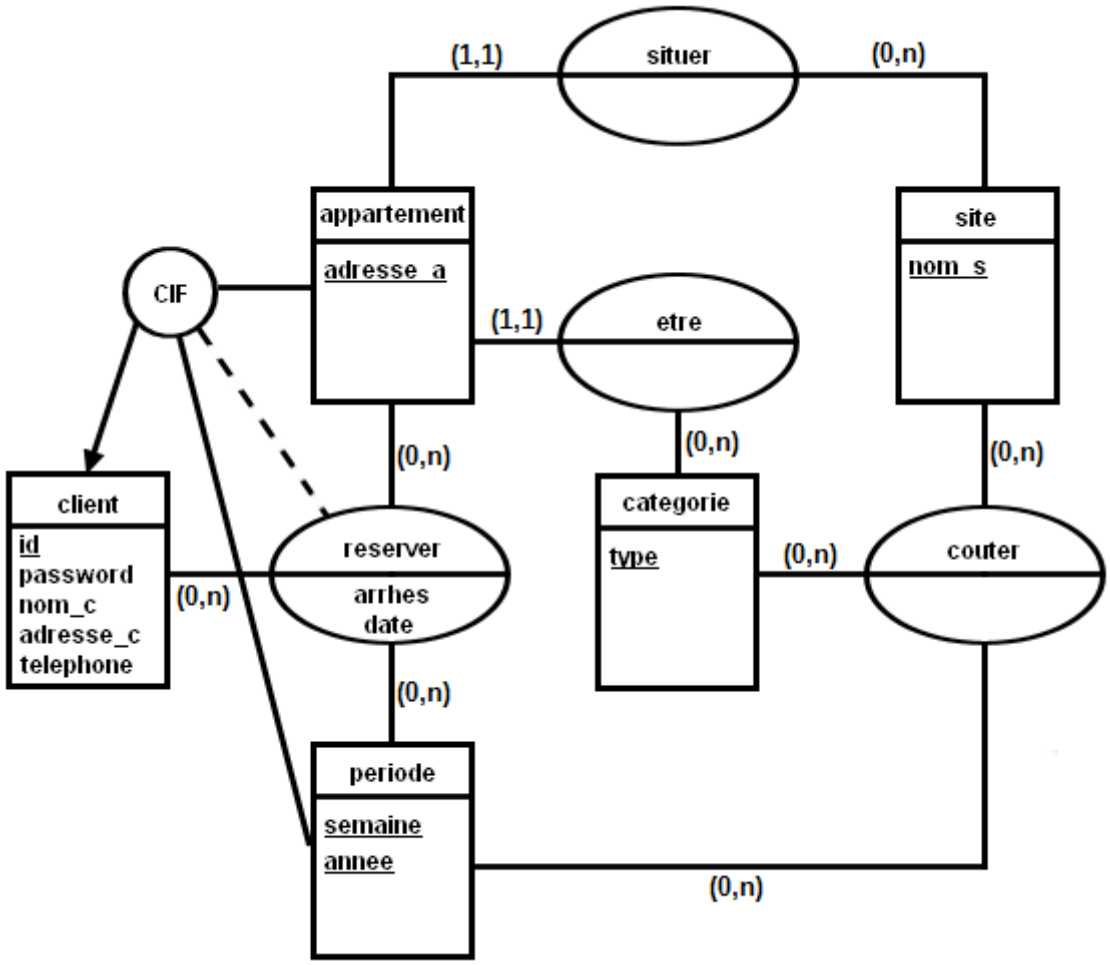

Figure 1 : Modèle de la base de données

1.1 Les entités

1.1.1 L'entité client

Cette entité regroupe les différents clients de la base de données. Elle possède cinq attributs, qui sont un identifiant, un mot de passe, un nom, une adresse et un numéro de téléphone (figure 2). L'identifiant est attribué au client lorsque ce dernier s'inscrit sur l'interface de réservation. C'est un nombre entier naturel non nul propre à chaque client, qui permet d'identifier chacun d'entre eux. Par conséquent, cet attribut est la clé primaire de l'entité. Le mot de passe est obligatoire et choisi par le client lorsqu'il s'inscrit. C'est une chaîne de caractères d'au plus 15 éléments. Le client a la possibilité de modifier son mot de passe s'il le désire. Le nom est le nom du client, que ce dernier est dans l'obligation de déclarer lors de son inscription. C'est une chaîne de caractères. Il n'est pas possible de la modifier. L'adresse est l'adresse actuelle du client. Cette dernière est obligatoire, et le client doit l'indiquer lors de son inscription. C'est une chaîne de caractères pouvant être modifiée par le client. Le numéro de téléphone est un entier naturel non nul. Tout comme l'adresse, il est obligatoire et doit être indiqué lorsque le client s'inscrit. Il est également possible de le modifier.

Figure 2 : Entité client

1.1.2 L'entité appartement

Cette entité regroupe les différents appartements appartenant à l'agence immobilière. Elle possède un unique attribut : l'adresse de l'appartement (qui comprend le numéro de l'appartement, le nom du bâtiment auquel il appartient et l'adresse complète de la rue où il est situé) (figure 3). Cet attribut est bien unique pour chaque appartement, ce qui justifie qu'il soit la clé primaire de l'entité. C'est une chaîne de caractères.

Figure 3 : Entité appartement

1.1.3 L'entité site

Entité qui contient les différents sites de la station. Il y a un unique attribut nom, qui correspond au nom des sites (figure 4). Chaque site a un nom qui lui est propre, donc l'attribut nom est la clé primaire de l'entité. C'est un attribut qui ne peut prendre que certaines valeurs (disponibles dans la liste des noms de sites existants).

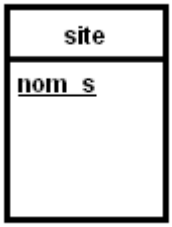

Figure 4 : Entité site

1.1.4 L'entité categorie

Cette entité correspond à l'ensemble des catégories d'appartements disponibles et possède un seul attribut (figure 5). Les noms des différentes catégories sont tous différents, ce qui justifie l'utilisation de cet attribut comme clé primaire de l'entité. L'attribut prend une valeur parmi la liste des catégories disponibles.

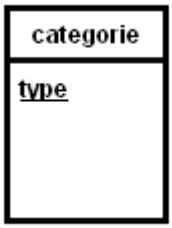

Figure 5 : Entité categorie

1.1.5 L'entité periode

Cette entité regroupe les périodes de réservation. Les locations se font à la semaine, par conséquent l'entité est composée de deux attributs : semaine et annee, qui correspondent respectivement au numéro de la semaine dans l'année et à l'année (figure 6). Le couple d'attributs forme la clé primaire de l'entité (il n'est pas possible de choisir qu'un seul de ses attribut comme clé primaire, car il y a plusieurs semaines dans une année et que le numéro d'une semaine est le même d'une année à l'autre). L'attribut semaine est un entier naturel non nul compris entre 1 et 53 (chaque année est composée de 52 ou 53 semaines). L'attribut annee est une année.

Figure 6 : Entité periode

1.2 Les associations

1.2.1 L'association reserver

Cette association modélise les réservations d'appartements. Comme un client fait une demande de réservation pour un appartement, à une période donnée, l'association est ternaire et relie les trois entités suivantes : client, appartement et periode (figure 7). Chaque demande de réservation donne lieu à la création d'un dossier, identifiable par son numéro (qui est donc propre à chaque demande). Comme un même appartement ne peut pas faire l'objet de plusieurs réservations sur une même période, le numéro de dossier est composé de l'adresse de l'appartement et du couple semaine et année. La clé primaire de l'association est donc la concaténation des clés primaires des entités appartement et periode. Concernant la clé primaire de l'entité client, elle est une clé étrangère de l'association. Cette association possède deux attributs propres : arrhes et date. L'attribut arrhes indique si les arrhes ont été payées ou non (les arrhes représentant 10% du montant total). La valeur de cet attribut doit obligatoirement être indiquée (et cette dernière est comprise dans un domaine à deux éléments). Concernant l'attribut date, il représente la date à laquelle la demande a été réalisée. Cet attribut est également obligatoire, car si les arrhes n'ont pas étés payées dans les quinze jours qui suivent la demande de location, la réservation est annulée.

De plus, un appartement ne peut avoir que trois états possibles : soit il est libre, soit il est bloqué, soit il est réservé de manière définitive. S'il n'est pas réservé, alors il est libre. Sinon, soit les arrhes ont été payées et l'appartement est définitivement réservé, soit elles n'ont pas été réglées et l'appartement est seulement bloqué. L'état d'un appartement se déduit donc à partir d'autres attributs, il n'est pas nécessaire de stocker l'état dans un nouvel attribut.

Pour les cardinalités, elles sont toutes en zéro à plusieurs. En effet, un appartement peut ne pas être réservé ou bien être réservé plusieurs fois (à différentes périodes). Plusieurs appartements peuvent être réservés la même période tout comme une période peut n'avoir aucune réservation. Concernant un client, il a la possibilité de n'avoir aucune réservation en cours ou d'en avoir plusieurs. Cependant, un client ne peut pas avoir plus de trois réservations non validées (ie dont les arrhes n'ont pas encore été payées).

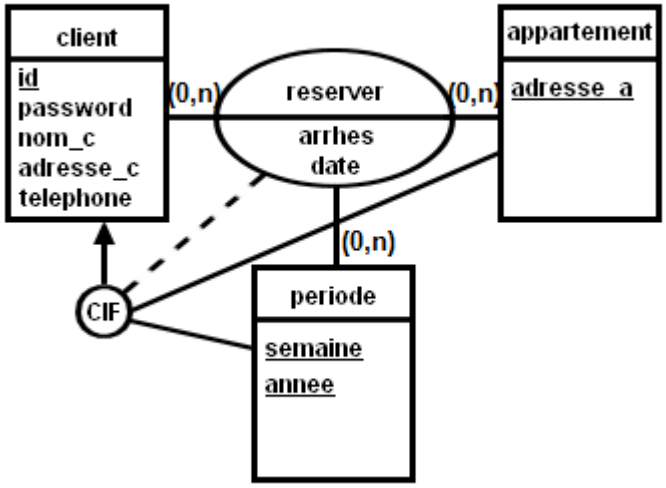

Figure 7 : Association reserver

1.2.2 L'association etre

C'est une association qui indique le type d'un appartement. Elle est binaire et relie les entités appartement et categorie (figure 8).

Comme un appartement est d'un et un seul type, sa cardinalité est d'un à un. Par contre, il y a plusieurs appartements d'une même catégorie (voire aucun appartement), la cardinalité de l'entité catégorie pour cette association est donc zéro à plusieurs.

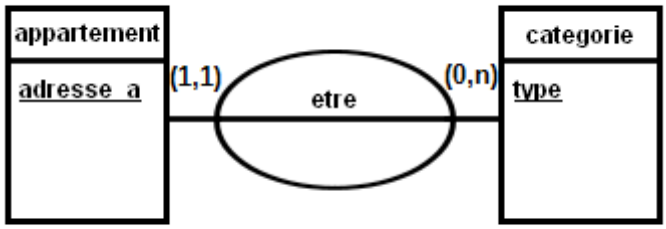

Figure 8 : Association etre

1.2.3 L'association situer

L'association situer indique qu'un appartement est présent sur un des sites de la station. C'est donc une association binaire entre les entités appartement et site (figure 9).

De la même manière que l'association précédente, la cardinalité de l'entité appartement pour cette association est d'un à un car un appartement est situé sur un et un seul site. Quant à un site, il peut contenir aucun appartement ou plusieurs, l'entité site a donc une cardinalité en zéro à plusieurs.

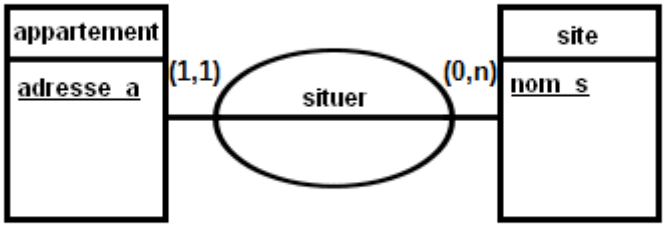

Figure 9 : Association situer

1.2.4 L'association couter

Le prix de la location dépend de trois paramètres : le type de l'appartement loué, le site sur lequel il se trouve et la période de l'année pendant laquelle il est réservé. L'association couter est donc ternaire (entre les entités categorie, site et periode) (figure 10). On suppose que le prix est le même pour tout appartement de catégorie identique situé sur un même site pour une même période, il n'est donc pas nécessaire de stocker le prix dans un attribut (puisqu'il se déduit des attributs des entités en association).

Les cardinalités sont toutes en zéro à plusieurs. En effet, les trois catégories peuvent ne rien coûter (s'il n'existe pas d'appartement d'un certain type, ou si un site ne dispose d'aucun appartement ou encore si l'agence décide de fermer la location à une période donnée) ou avoir des coûts différents (le coût étant une fonction de trois paramètres, il suffit d'en faire varier un seul pour modifier le prix).

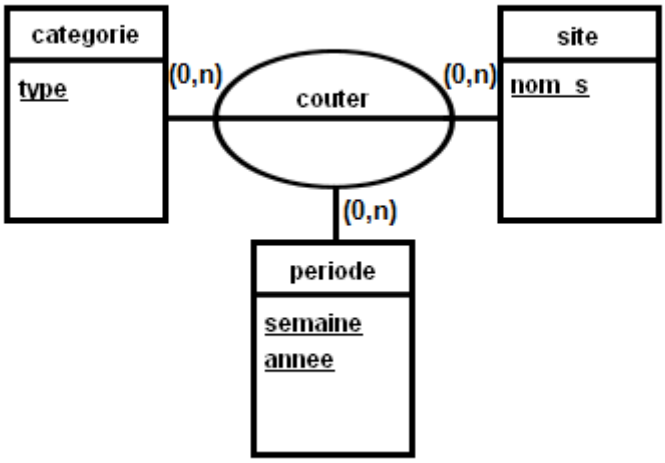

Figure 10 : Association couter

1.3 Le modèle relationnel

Les entités deviennent des relations et gardent l'ensemble de leurs attributs. Les clés de chaque entité deviennent clés primaires de la relation correspondante. Les associations etre et situer ont une cardinalité maximale à 1. Par conséquent, elle ne deviennent pas des relations et la relation appartement reçoit comme attribut les clés des relations site et categorie. Les associations couter et reserver n'ont aucune cardinalité maximale à 1, ces associations deviennent des relations. On obtient le modèle relationel suivant :

client(id, password, nom_client, adresse_client, telephone)
appartement(adresse_appart, site_nom_site, categorie_type)
site(nom_site)
categorie(type)
periode(semaine, annee)
couter(site_nom_site, categorie_type, periode_semaine, periode_annee)
reserver(client_id, appartement_adresse_appart, periode_semaine, periode_annee, arrhes, date)

2 La modélisation

La modélisation du modèle relationnel défini précédemment est réalisée en MySQL via l'outil MySQL Workbench (figure 11).

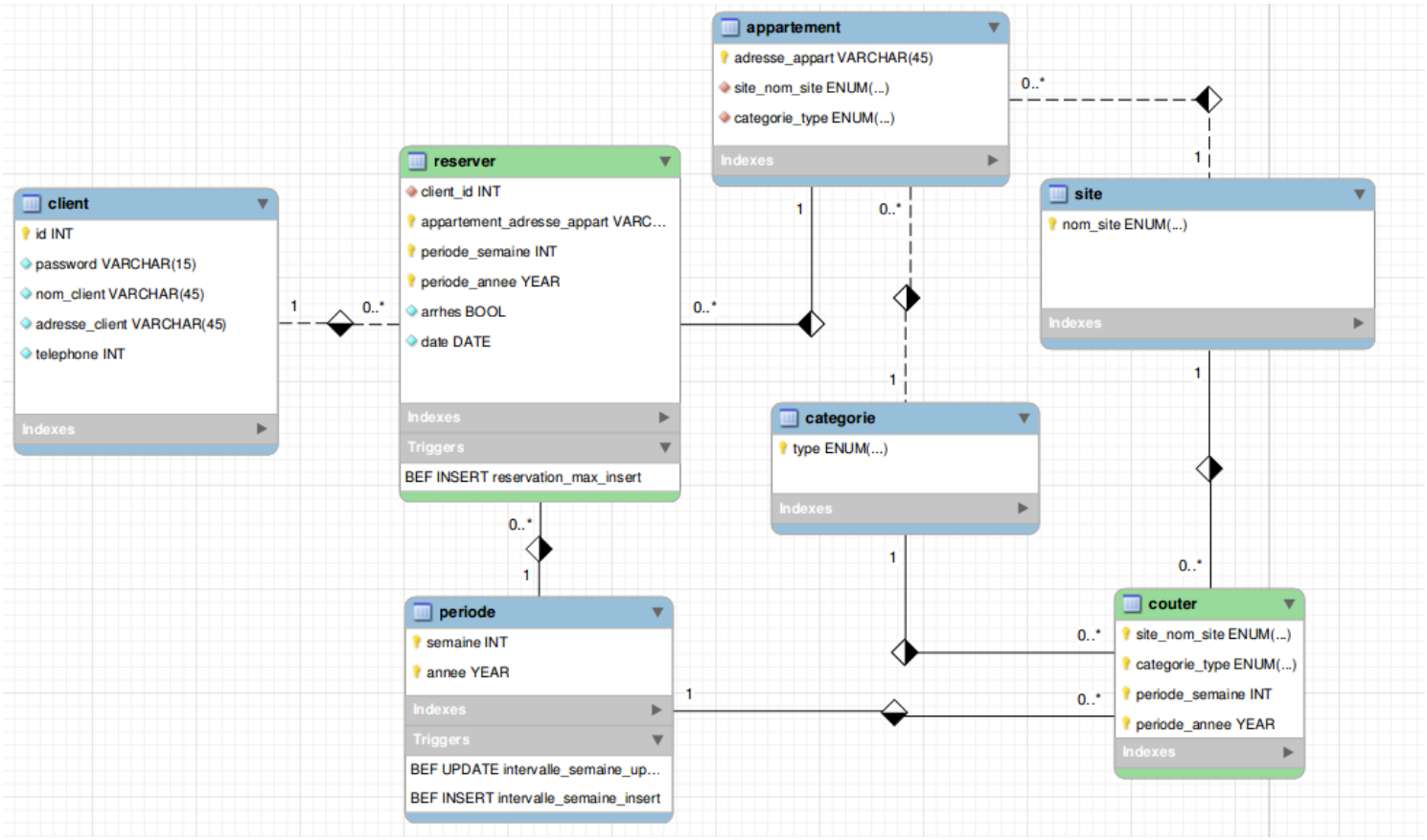

Figure 11 : Modélisation du modèle relationnel à l'aide de MySQL Workbench

2.1 Les domaines

Lorsque l'attribut est un nombre, on choisit de le représenter par un entier naturel. Le domaine choisi est donc un entier non signé (INT UN). Pour les attributs qui sont des chaînes de caractères, le domaine retenu est VARCHAR. Les chaînes sont limitées à 45 caractères pour les adresses et le nom des clients est limité à 15 pour les mots de passe. Le type de l'attribut annee est YEAR et celui de date est DATE. L'attribut arrhes ne peut prendre que deux valeur (payée ou non payée), on le représente donc par un booléen (BOOL). Concernant les deux attributs qui ne peuvent prendre leur valeur que dans une liste prédéfinie (les attributs nom_site et type), les domaine utilisés sont des énumérations des valeurs possibles (ENUM). De plus, le nombre de semaines par an est au plus de 53, on utilise alors un trigger pour refuser toute insertion ou mise à jour de cet attribut avec une valeur qui n'est pas comprise entre 1 et 53.

2.2 La contrainte de cardinalité de la relation client

Un client ne peut pas faire plus de trois demandes non validées, ie qu'il ne peut pas y avoir plus de trois n-uplet dans la table reserver dont l'attribut client_id est le même et l'attibut arrhes est FALSE. Lorsqu'un client fait une demande, un trigger sur la table reserver vérifie le nombre de demandes non validées de ce client. Si elles sont au nombre de 3, l'insertion d'une nouvelle demande de réservation est refusée et un message d'erreur est retourné.

2.3 Les clés étrangères

On choisit de rajouter l'option ON DELETE CASCADE à l'ensemble des clés étrangères du modèle pour conserver une cohérence lors de suppressions.

2.4 Les routines

2.4.1 Les routines client

On regroupe les routines concernant les modifications de la table client. Ces dernières sont au nombre de trois, et sont toutes des procédures : inscription, connexion et modification_informations (figure 12).

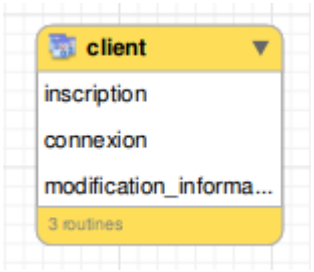

Figure 12 : Les routines client

Lorsqu'un client s'inscrit, il indique son nom, son adresse, son numéro de téléphone et se choisit un mot de passe. Le serveur de traitement appelle la procédure inscription qui insère ces différentes informations, ainsi qu'un identifiant unique, dans la table client (toutes les données sont donc passées en paramètre d'entrée de la procédure). L'identifiant est une variable globale, @new_id (on connaît le nom de la variable globale et on utilise toujours cette dernière donc on décide de ne pas la passer en paramètre), qui est communiquée au client une fois l'insertion effectuée. Cette variable est ensuite incrémentée de un, ce qui permet d'assurer son unicité.

Quand un client est inscrit, il se connecte au serveur de traitement à l'aide de son identifiant et de son mot de passe. Il rentre ces deux informations dans les champs correspondants, puis le serveur appelle la procédure connexion. Cette dernière prend les deux données en paramètre d'entrée puis lance une requête qui compte le nombre de n-uplets présents dans la base dont les valeurs des attributs id et password sont égales aux paramètres. Le résultat ne peut être que 1 ou 0 (car l'attribut id est clé primaire et donc unique) et si ce dernier vaut 0, un message d'erreur est retourné.

On donne le droit au client de modifier son mot de passe, son adresse et son numéro de téléphone (ce qui est justifié pour les cas d'oubli de mot de passe ou les changement de domicile). Le serveur propose au client, lorsque ce dernier est connecté, de modifier ses informations. Le client à l'obligation d'indiquer l'ensemble des informations, même celles inchangées. Le serveur de traitement appelle ensuite la procédure modification_informations qui prend en paramètre d'entrée les informations. Cette dernière réalise une mise à jour du n-uplet de la table client dont l'attribut id correspond avec celui utilisé lors de la connexion.

2.4.2 Les routines de réservation

On décide de séparer les routines concernant la modification de la table reserver. On regroupe les procédures n'ayant pas de lien avec le prix de la réservation. Il y en a cinq : demande_reservation, choix_reservation, location_terminee, consultation_reservations et suppression_reservation (figure 13).

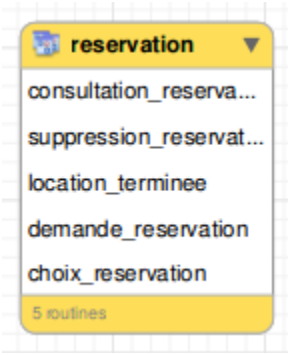

Figure 13 : Les routines de réservation

Les demandes de réservation sont traitées de la manière suivante : si un appartement de la catégorie désirée est disponible sur le site désiré et pour la période désirée, alors le client peut confirmer la demande. Si la demande ne peut pas être satisfaite, une solution de remplacement est proposée. Dans un premier temps, on propose au client les appartements disponibles sur le site désiré mais dans une autre catégorie. Puis, s'il n'y a pas d'appartements disponibles sur ce site, alors

on propose les appartements disponibles de la catégorie désirée présents sur les autres sites. Si ces solutions de remplacement ne sont pas possibles, un message de rejet est envoyé au client. On a choisit de modéliser les demandes de réservation via l'appel de deux procédures : demande_reservation et choix_reservation. La première vérifie la disponibilité des appartements (en fonction des critères désirés), puis les affiche quand ces derniers sont disponibles. Pour vérifier la disponibilité d'appartements, on réalise la différence entre l'ensemble des appartements dont les caractéristiques sont celles désirées et l'ensemble des ces appartements qui sont déjà réservés. Pour cela, on compte le nombre de n-uplets de la table appartement dont les attributs site_nom_site et categorie_type correspondent aux valeurs désirées mais dont l'attribut adresse_appart n'est pas égal à l'attribut appartement_adresse_appart d'un n-uplet de la table reserver (dont les attributs periode_semaine et periode_annee correspondent aux valeurs de la période désirée). Si ce nombre de n-uplets est différents de 0, alors la procédure sélectionne l'adresse ces appartements (ie l'attribut adresse_appart de la table appartement), sinon elle teste le nombre de n-uplets sans tenir compte de l'attribut categorie_type. De la même façon, si ce nombre est différent de 0, la procédure sélectionne l'adresse de ces appartements, sinon la procédure compte le nombre d'appartements disponibles sans tenir compte du site et les affiche s'il y en a. Sinon, la procédure renvoie un message d'erreur. Concernant la procédure choix_reservation, elle prend en paramètre d'entrée l'identifiant du client, l'adresse de l'appartement réservé et la période de réservation (la date de réservation est une variable globale appelée @date_actuelle, on connaît le nom de cette variable et on utilise toujours cette dernière donc on décide de ne pas la passer en paramètre). La procédure insère un n-uplet dans la table reserver dont la valeur de l'attribut arrhes est FALSE et les valeurs des autres attributs sont celles des paramètres d'entrée. Ensuite, la procédure renvoie le n-uplet du produit cartésien des tables reserver et appartement pour récapituler l'ensemble des caractéristiques de la réservation.

Une fois que la location est finie, il n'est pas nécessaire de la conserver dans la table reserver. Comme les locations se font à la semaine, le serveur appelle la procédure location_terminee hebdomadairement. Cette dernière réalise la suppression de tout les n-uplets de la table reserver dont l'attribut periode_annee est strictement inférieur à la variable globale @annee_actuelle (qui s'incrémente toutes les années) ainsi que ceux dont l'attribut periode_annee est égal à @annee_actuelle et l'attribut periode_semaine est strictement inférieur à la variable globale @periode_semaine (qui s'incrémente toute les semaines et avec une réaffectation de sa valeur à 1 chaque année). On connaît les noms des deux variables globales et on utilisera toujours ces dernières donc on décide de ne pas les passer en paramètre.

On donne la possibilité au client de consulter ses demandes de réservation et de les supprimer s'il le désire. Dans ces cas là, le serveur appelle respectivement les procédures consultation_reservations et suppression_reservation. La première prend en paramètre l'identifiant du client (id) et réalise une requête qui sélectionne l'ensemble des n-uplets de la table reserver dont l'attribut client_id correspond au paramètre. Concernant la procédure suppression_reservation, le paramètre d'entrée est le numéro de dossier (ie l'adresse de l'appartement et la période de réservation). Le n-uplet de la table reserver, dont les attributs appartement_adresse_appart, periode_semaine et periode_annee correspondent au numéro de dossier, est supprimé.

2.4.3 Les routines de paiement

Ces routines concernent tout ce qui est lié au prix de la réservation et du paiement des arrhes. Il y en a quatre, deux fonctions (calcul_prix et calcul_prix_arrhes) et deux procédures (arrhes_paiement et arrhes_impayees) (figure 14).

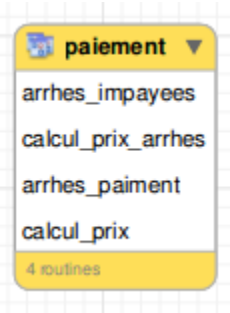

Figure 14 : Les routines de paiement

Le prix étant déterminer à partir du type de l'appartement, du site où il est situé ainsi que de la période de réservation, on utilise une fonction, calcul_prix, qui prend en paramètre l'adresse d'un appartement et la période. Dans la fonction, on déclare trois variables : prix, site et type. Les variables site et prix sont affectées respectivement avec la valeur de l'attribut site_nom_site et la valeur de l'attribut categorie_type de la table appartement dont l'attribut adresse_appart correspond à l'adresse passée en paramètre. Puis en fonction des valeurs des variables site, type, semaine et annee, le prix prend une certaine valeur (les tests sont réalisés par des CASE). La fonction retourne le prix.

Pour qu'une réservation soit définitive, le client doit payer les arrhes qui représentent 10 % du prix. Pour connaître ce montant, on décide d'implémenter une fonction calcul_prix_arrhes dont les paramètres sont les mêmes que ceux de la fonction calcul_prix. Une variable est déclarée, et on affecte à cette dernière la valeur de la fonction calcul_prix multipliée par (10/100). La variable est ensuite retournée.

Quand le client a effectué le paiement des arrhes, la procédure arrhes_paiement modifie l'attribut arrhes de la table reserver dont les attribut appartement_adresse_appart, periode_semaine et periode_annee correspondent au numéro de dossier passé en paramètre d'entrée (la valeur de arrhes passe de FALSE à TRUE).

Si le client n'a pas payé les arrhes de sa demande de réservation sous quinze jours, cette dernière est supprimée. Pour cela, le serveur de traitement appelle quotidiennement la procédure arrhes_impayees qui supprime tous les n-uplet de la table reserver dont l'attribut arrhes vaut FALSE si la différence de la variable globale @date_actuelle (comme pour la procédure choix_reservation, on décide de ne pas passer cette variable en paramètre car c'est toujours cette dernière qui est utilisée) et de l'attribut date de ces n-uplets est strictement supérieure à 15.

3 La simulation

La base de données modélisée via MySQL Workbench contient plusieurs données de référence. Pour vérifier le bon fonctionnement de la modélisation, en particulier les routines implémentées, on simule l'interaction entre un nouveau client, le serveur de traitement et le serveur de données. Les interactions entre le client et le serveur de traitement seront simulées par du texte, et les interactions entre les serveurs de traitement et de données seront simulées à l'aide de commandes MySQL.

3.1 L'inscription et la connexion

Dans un premier temps, on simule l'inscription d'un client. On suppose que ce dernier se déconnecte puis essaye de se reconnecter. On vérifie qu'un message d'erreur est bien envoyé lorsque le client se trompe d'identifiant ou de mot de passe. On simule ensuite le changement d'informations du client.

3.2 Les demandes de réservation

On suppose que c'est le début de la semaine et une des réservations enregistrée dans la base vient de se finir. On vérifie que cette dernière est bien supprimée de la table reserver.

Le client réalise plusieurs demandes de réservation d'appartement. Les demandes concernent des appartements de types différents, situés sur des sites différents et sur différentes périodes. Tout les cas sont traités : la première demande est satisfaite, la deuxième demande ne peut pas être satisfaite, la troisième demande utilise la première solution de remplacement (ie un appartement sur le site désiré mais d'un type différent) et la quatrième demande utilise la deuxième solution de remplacement (ie un appartement dans la catégorie demandée mais sur un autre site). Lorsque les demandes peuvent être satisfaites, le client choisit à chaque fois l'appartement qu'il désire louer.

Le client souhaite ensuite réaliser une quatrième demande, qui sera refusée car il a déjà trois demandes non validées. Il va alors afficher l'ensemble de ses demandes et en supprimer une. On vérifie que la demande a bien été supprimée (le client visualise de nouveau ses réservations). Puis on simule de nouveau la demande pour vérifier que, cette fois-ci, elle a bien été prise en compte.

3.3 Le paiement des arrhes

Le client souhaite effectuer le paiement des arrhes d'une de ses réservations. On vérifie que le prix des arrhes à payer est bien communiqué et que lorsque le paiement est fait, la demande de réservation devient valide (ie que l'attribut arrhes de ce n-uplet passe à TRUE). On souhaite ensuite vérifier que les réservations non valides sont bien supprimées de la table lorsque quinze jours se sont écoulés après la date de la demande. Pour ce faire, on simule l'appel de la procédure de suppression après quatorze jours (pour s'assurer qu'il n'y a pas de suppression) puis après quinze jours (pour s'assurer que cette fois-ci, la suppression a eu lieu).

Après avoir défini un modèle relationnel pour une agence immobilière effectuant des location d'appartements dans une station de ski, on l'a modélisé en MySQL via l'outil Workbench. On a introduit différents triggers et routines pour rendre la modélisation utilisable. On a enfin effectué une simulation d'interaction entre un client, le serveur de traitement et la base de données pour vérifier que les implémentations étaient correctes.