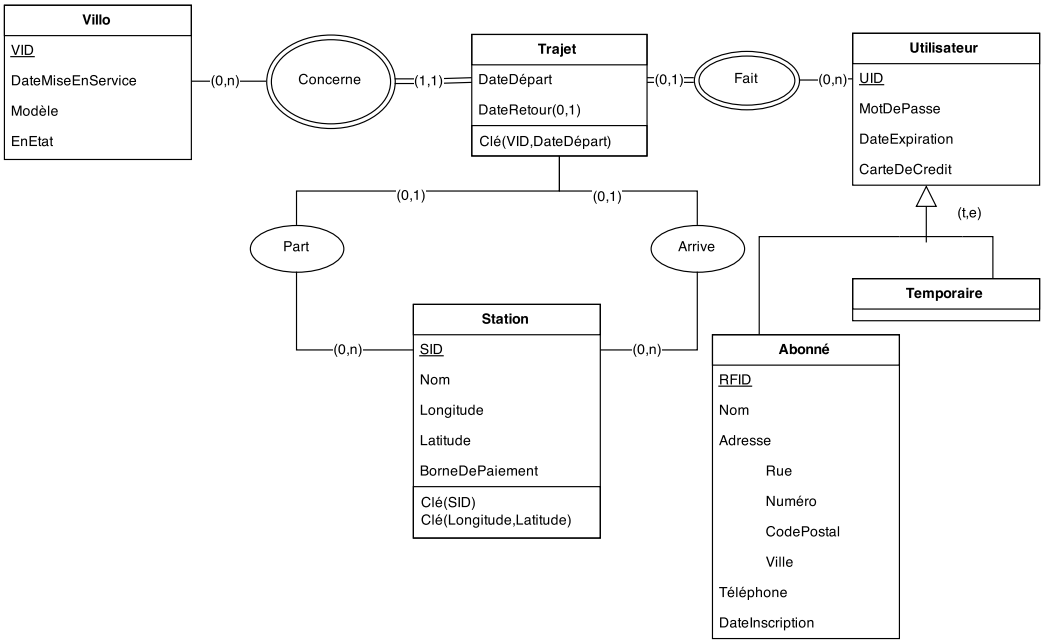
Projet INFO-H-303 : Villo!

Hereman Nicolas, Van Brande Rodrigue

24 avril 2015

Diagramme entité-association



Contraintes et hypothèses

- La *DateExpiration* d'un *Abonné* doit être strictement supérieure à sa *DateInscription*
- La *DateDépart* d'un *Trajet* doit être strictement supérieure à la *DateInscription* de l'*Abonné* qui le fait.
- La *DateDépart* d'un *Trajet* pot être strictement inférieure à la *DateExpiration* de l'*Utilisateur*.
- La *DateDépart* d'un *Trajet* doit être strictement inférieure à la *DateExpiration* de l'*Utilisateur* qui le fait.
- La *DateRetour* d'un *Trajet* doit être strictement supérieure à la *DateDépart* de ce même *Trajet*.
- La *DateDépart* d'un *Trajet* doit être strictement supérieure à la *DateMiseEnService* du *Villo* concerné.
- A l'instant *DateRetour*, la *Station* de retour d'un *Trajet* doit contenir moins de *Villos* que sa *Capacité*. Le calcul du nombre de *Villos* présents dans une *Station* à l'aide de l'entité *Trajet*.
- Un même *Villo* ne peut pas concerner deux *Trajets* en même temps.
- Un même *Utilisateur* ne peut pas faire deux *Trajets* en même temps.
- Un *Trajet i* qui suit directement un autre *Trajet j* pour un même *Villo* doit avoir la même *Station* de départ que la *Station* d'arrivée du *Trajet j*.
- Si un *Trajet* n'a pas d'*Utilisateur*, il n'a pas de *Station* de départ et inversement.
- Si un *Trajet* n'a ni *Utilisateur* ni *Station* de départ, sa *DateDépart* est 0000/00/00 - 00:00:00
- Pour chaque *Villo*, il n'y a qu'un *Trajet* sans *Utilisateur* et *Station* de départ. Ni plus ni moins.
- Si un *Trajet* n'a pas de *Station* de retour, il n'a pas de *DateRetour* et inversement.
- Pour chaque *Villo*, il y a au maximum un *Trajet* sans *DateRetour* et *Station* de retour.

- Si un *Trajet* n'a pas de *DateDépart*, il a une *DateRetour*.
- Si un *Trajet* n'a pas de *DateRetour*, il a une *DateDépart*.

Traduction relationnelle

- Utilisateur(UID,MotDePasse,DateExpiration,CarteDeCredit)
- Abonne(UID, RFID, Nom, Rue, Numéro, CodePostal, Ville, Téléphone, DateInscription)
  - UID référence Utilisateur.UID
- Station(SID, Nom, Longitude, Latitude, Capacité, BorneDePaiement)
- Villo(VID,DateMiseEnService,Modèle, EnEtat)
- Trajet(VID,DateDépart, *UID*, *StationDépart*, *DateRetour*, *StationRetour*)
  - UID référence Utilisateur.UID
  - VID référence Villo.VID
  - StationDépart référence Station.SID
  - StationRetour référence Station.SID

Remarque

Un Utilisateur.UID n'existant pas dans Abonné.UID est un utilisateur temporaire

Justification et hypothèses de modélisation

On utilise une table *Utilisateur* pour stocker leurs données. On a créé une table *Abonné* afin des les différencier des utilisateurs *Temporaire*. Comme l'héritage des table est totale et exclusive et que toutes les informations dont on a besoin pour ces derniers sont déjà dans la table *Utilisateur*, ils n'ont pas besoin d'une table pour eux. Les utilisateurs temporaires seront ceux qui n'ont pas leurs *UID* dans la table *Abonné*.

Les *Stations* et *Villos* ont aussi droit à leur table pour qu'on y sauvegarde leurs données.

On sauvegarde aussi la liste des *Trajets* dans une table. Le fait que l'*Utilisateur* et la *StationDépart* soit optionnel peut paraître illogique par rapport à la réalité. Mais cela se justifie par le fait que les *Villo* doivent être placé une première fois. Comme on ne sauvegarde pas la *Station* dans laquelle ils sont stockés, on se repère au *Trajet* pour le savoir. On les place donc à l'aide de *Trajet* sans *Station* de départ et sans *Utilisateur*. Les *Trajets* sans *Station* de retour et *DateRetour* sont les *Trajets* encore en cours.

Script SQL DDL de création de base de données

```
1  CREATE TABLE `Utilisateur` (
2          `UID` int unsigned NOT NULL,
3          `MotDePasse` smallint(4) unsigned zerofill NOT NULL,
4          `CarteDeCredit` bigint(16) unsigned zerofill NOT NULL,
5          `DateExpiration` datetime NOT NULL,
6          PRIMARY KEY (`UID`)
7  );
8
9  CREATE TABLE `Villo` (
10         `VID` smallint unsigned NOT NULL,
11         `DateMiseEnService` datetime NOT NULL,
12         `Modèle` varchar(12) NOT NULL,
13         `EnEtat` tinyint(1) NOT NULL,
14         PRIMARY KEY (`VID`)
15 );
16
17 CREATE TABLE `Abonné` (
18         `UID` int unsigned NOT NULL,
19         `RFID` char(20) NOT NULL,
20         `Nom` varchar(50) NOT NULL,
21         `Rue` varchar(100) NOT NULL,
22         `Numéro` smallint unsigned NOT NULL,
23         `CodePostal` smallint(4) unsigned NOT NULL,
24         `Ville` varchar(50) NOT NULL,
25         `Téléphone` char(10) NOT NULL,
26         `DateInscription` datetime NOT NULL,
27         PRIMARY KEY (`RFID`),
28         FOREIGN KEY (`UID`) REFERENCES Utilisateur(`UID`)
29 );
30
```

```
31 CREATE TABLE `Station` (
32         `SID` smallint unsigned NOT NULL,
33         `Nom` varchar(50) NOT NULL,
34         `Longitude` float NOT NULL,
35         `Latitude` float NOT NULL,
36         `Capacité` tinyint unsigned NOT NULL,
37         `BorneDePaiement` tinyint(1) NOT NULL,
38         PRIMARY KEY (`SID`,`Longitude`,`Latitude`)
39 );
40
41 CREATE TABLE `Trajet` (
42         `VID` smallint unsigned NOT NULL,
43         `DateDépart` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
44         `UID` int unsigned DEFAULT NULL,
45         `StationDépart` smallint unsigned DEFAULT NULL,
46         `DateRetour` datetime DEFAULT NULL,
47         `StationRetour` smallint unsigned DEFAULT NULL,
48         PRIMARY KEY (`VID`,`DateDépart`),
49         FOREIGN KEY (`VID`) REFERENCES Villo(`VID`),
50         FOREIGN KEY (`UID`) REFERENCES Utilisateur(`UID`),
51         FOREIGN KEY (`StationDépart`) REFERENCES Station(`SID`),
52         FOREIGN KEY (`StationRetour`) REFERENCES Station(`SID`)
53 );
```

Les requêtes en algèbre relationnel et calcul relationnel tuples

Requêtes demandées

R1

Les utilisateurs habitant Ixelles ayant utilisé un Villo de la station Flagey

Algèbre relationnelle:

$UTS \leftarrow (Utilisateur * Trajet) \bowtie_{StationDpart=SID} Station$

$Sel \leftarrow \sigma_{Station.Nom=Flagey \wedge Utilisateur.Ville=Ixelles}(UTS)$

$\pi_{Utilisateur.UID}(Sel)$

Calcul relationnel tuple:

$$\{u.UID | Utilisateur(u) \wedge u.Ville = Ixelles \wedge \exists t \exists s(Trajet(t) \wedge Station(s) \wedge t.UID = u.UID \wedge s.Nom = Flagey)\}$$

Imprécisions:
L'énoncé ne dit pas si il faut récupérer l'UID où le nom de l'utilisateur. Nous avons donc supposé qu'il s'agissait de l'UID.

R2

Les utilisateurs ayant utilisé Villo au moins 2 fois

Algèbre relationnelle:
TODO

Calcul relationnel tuple:

$$\{u.UID | Utilisateur(u) \wedge \exists t1 \exists t2(Trajet(t1) \wedge Trajet(t2) \wedge t1.UID = u.UID \wedge t2.UID = u.UID \wedge [t1.VID \neq t2.VID \vee t1.DateDepart \neq t2.DateDepart])\}$$

Imprécisions:
TODO

R3

Les paires d'utilisateurs ayant fait un trajet identique

Algèbre relationnelle:
$t2(UID2, dep2, ret2) \leftarrow \pi_{UID, StationDpart, StationRetour}(Trajet)$
$Temp \leftarrow Trajet \bowtie_{StationDpart=dep2 \wedge StationRetour=ret2} t2$
$\pi_{UID, UID2}(Temp)$

Calcul relationnel tuple:

$$\{u1.UID, u2.UID | Utilisateur(u1) \wedge Utilisateur(u2) \wedge \exists t1 \exists t2(Trajet(t1) \wedge Trajet(t2) \wedge t1.UID = u1.UID \wedge t2.UID = u2.UID \wedge t1.StationDepart = t2.StationDepart \wedge t1.StationRetour = t2.StationRetour)\}$$

Imprécisions:
Voir *R1*.

R4

Les vélos ayant deux trajets consécutifs disjoints (station de retour du premier trajet différente de la station de départ du suivant).

Algèbre relationnelle:
TODO

Calcul relationnel tuple:
TODO

Imprécisions:
TODO