

UNIVERSITÉ LIBRE DE BRUXELLES

Rapport : Villo !

Pierre Gérard, Titouan Christophe

INFO-H-303 Base de données

Esteban Zimányi, Michaël Waumans

Table des matières

1	Diagramme entité association	2
1.1	Diagramme	2
1.2	Contraintes	2
2	Modèle relationnel	2
2.1	Modèle	2
2.2	Contraintes	3
3	Hypothèses sur le modèle	3
4	Justification	3
5	Requêtes	4
5.1	Les utilisateurs habitant Ixelles ayant utilisé un Villo de la station Flagey	4
5.1.1	SQL	4
5.1.2	Algèbre relationnelle	4
5.1.3	Calcul relationnel	4
5.2	Les utilisateurs ayant utilisé Villo au moins 2 fois	4
5.2.1	SQL	4
5.2.2	Algèbre relationnelle	4
5.2.3	Calcul relationnel	5
5.3	Les paires d'utilisateurs ayant fait un trajet identique	5
5.3.1	SQL	5
5.3.2	Algèbre relationnelle	5
5.3.3	Calcul relationnel	5
5.4	Les vélos ayant deux trajets consécutifs disjoints (station de retour du premier trajet différente de la station de départ du suivant)	5
5.4.1	SQL	5
5.4.2	Algèbre relationnelle	6
5.4.3	Calcul relationnel	6
5.5	Les utilisateurs, la date d'inscription, le nombre total de trajet effectués, la distance totale parcourue et la distance moyenne parcourue par trajet, classés en fonction de la distance totale parcourue	6
5.6	Les stations avec le nombre total de vélos déposés dans cette station (un même vélo peut-être comptabilisé plusieurs fois) et le nombre d'utilisateurs différents ayant utilisé la station et ce pour toutes les stations ayant été utilisées au moins 10 fois.	6

1 Diagramme entité association

1.1 Diagramme

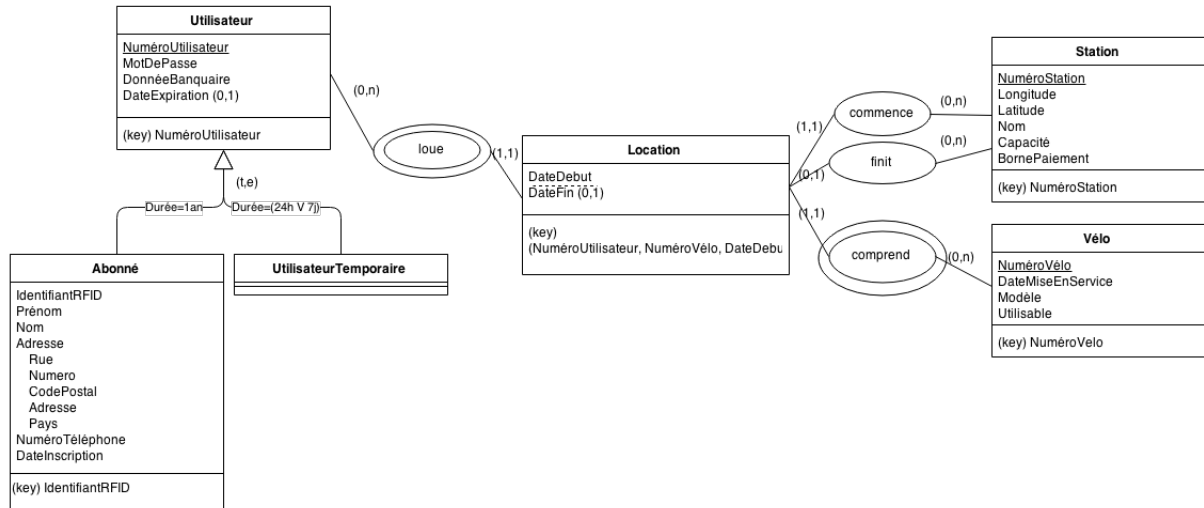


FIGURE 1 – Diagramme entité association

1.2 Contraintes

Les contraintes sont les suivantes :

- La DateDebut d'une Location doit précéder la DateFin,
- La DateMiseEnService d'un Vélo doit précéder la DateDebut de chacune de ses Locations,
- La DateExpiration d'un Utilisateur doit être postérieure à la DateDebut de toutes ses Locations,
- Le couple (Longitude, Latitude) est unique,
- La DateExpiration d'un Utilisateur doit être postérieure à la DateFin de toutes ses Locations,
- Un utilisateur qui a une date d'expiration non nul ne peut pas avoir plus d'une Location ayant une DateFin nul,
- Pour une Station, le nombre de vélo dont la dernière Location finit dans cette station ne doit pas dépasser sa capacité,
- Un Vélo ne peut pas avoir de déplacement disjoints. C'est à dire que la Station de départ du trajet n doit être similaire à la Station d'arrivée du trajet (n-1) pour $n \geq 1$,
- Un Utilisateur ayant une DateExpiration non nul ne peut pas prendre un Vélo si Usable est faux.

2 Modèle relationnel

2.1 Modèle

Utilisateur(NuméroUtilisateur, MotDePasse, DonnéeBanquaire, DateExpiration)

Abonné(NuméroUtilisateur, IdentifiantRFID, Nom, Rue, Numero, CodePostal, Adresse, Pays DateInscription, NuméroTéléphone)

Abonné.NuméroUtilisateur référence Utilisateur.NuméroUtilisateur

Location(NuméroUtilisateur, NuméroVélo, DateDebut, DateFin, NuméroStationDépart, NuméroStationFin)

Location.NuméroUtilisateur référence Utilisateur.NuméroUtilisateur

Location.NuméroVélo référence Vélo.NuméroVélo

Location.NuméroStationDépart référence Station.NuméroStation

Location.NuméroStationFin référence Station.NuméroStation

(NuméroUtilisateur, DateDebut) est unique

(NuméroVélo, DateDebut) est unique

Station(NuméroStation, Longitude, Latitude, Nom, Capacité, BornePaieement)

Vélo(NuméroVélo, DateMiseEnService, Modèle, Utilisable)

2.2 Contraintes

Les contraintes sont les suivantes :

- Une Location a au plus une station d'arrivé,
- Une Location a une et une seule Station de départ,
- Une Location a un et un seul Utilisateur,
- Une Location a un et un seul Vélo,
- La DateDebut d'une Location doit précéder la DateFin,
- La DateMiseEnService d'un Vélo doit précéder la DateDebut de chacune de ses Locations,
- Le couple (Longitude, Latitude) est unique,
- La DateExpiration d'un Utilisateur doit être postérieure à la DateDebut de toutes ses Locations,
- Un utilisateur qui a une date d'expiration non nul ne peut pas avoir plus d'une Location ayant une DateFin nul,
- Pour une Station, le nombre de vélo dont la dernière Location finit dans cette station ne doit pas dépasser sa capacité,
- Un Vélo ne peut pas avoir de déplacement disjoints. C'est à dire que la Station de départ du trajet n doit être similaire à la Station d'arrivé du trajet $(n-1)$ pour $n \geq 1$,
- Un Utilisateur ayant une DateExpiration non nul ne peut pas prendre un Vélo si Usable est faux.

3 Hypothèses sur le modèle

Il existe des utilisateur "admin", ce sont ces derniers et uniquement eux qui n'ont pas de date d'expiration.

Si un Abonné a son abonnement qui expire, il peut re-utiliser le NuméroUtilisateur et MotDePasse dans le futur, l'entité n'est pas supprimée.

Dans le cas ou des employés villo déplacent un vélo la nuit, alors ce déplacement doit être enregistré dans la base de donnée par un utilisateur admin.

Dans le cas ou un vélo serait cassé et devrait sortir du circuit de location, un Utilisateur admin vient le chercher et la Location ne finit jamais, c'est à dire pas de DateFin.

Dans le cas ou la société villo achète des nouveaux vélo et les met en circulation, un utilisateur admin fait une location de ce nouveau vélo qui a une date de départ égale à la date d'arrivée et une station de départ égale à la station d'arrivée.

Le champs MotDePasse contient un hash cryptographique du mot de passe et non le mot de passe lui-même.

4 Justification

Afin d'éviter la redondance et de garantir la cohérence du modèle, nous avons choisis de :

- Ne pas mettre d'attribut PlaceUtilisé dans Station,
- Ne pas mettre d'attribut Endroit dans Vélo,
- ect ...

En effet, ces informations peuvent être déduite de la suite de Location.

Pour obtenir une clé primaire à l'entité Location, nous avons rendu obligatoire le champs DateDebut et c'est pour cela que la mise en circulation et la mise a la retraite des vélos sont différentes.

Pour la généralisation nous avons choisis, une solution permettant d'avoir une relation a une seule table depuis la location et une solution permettant de mettre une contrainte d'existence sur tous les champs excepté la DateExpiration.

5 Requêtes

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

5.1.1 SQL

```

1 -- Les utilisateurs habitant Ixelles ayant utilise un Villo de la station Flagey

SELECT DISTINCT subscriber.user_id,
                subscriber.firstname,
                subscriber.lastname
6 FROM    subscriber,
          trip
          INNER JOIN station
                ON trip.departure_station_id = station.id
WHERE subscriber.user_id = trip.user_id
11 AND station.NAME = "FLAGEY"
AND subscriber.address_zipcode = 1050;

```

5.1.2 Algèbre relationnelle

$$\Pi_{subscriber.user_id, subscriber.firstname, subscriber.lastname}(\sigma_{station.name="FLAGEY" \wedge subscriber.address_zipcode=1050}($$

(1)

$$subscriber \bowtie_{subscriber.user_id=trip.user_id} (trip \bowtie_{trip.departure_station_id=station.id} station))$$

(2)

5.1.3 Calcul relationnel

$$\{sb.user_id, sb.firstname, sb.lastname | subscriber(sb) \wedge trip(tp) \wedge station(st) \wedge tp.departure_station_id = st.id$$

(3)

$$\wedge sb.user_id = tp.user_id \wedge st.NAME = "FLAGEY" \wedge tp.departure_station_id = st.id\}$$

(4)

5.2 Les utilisateurs ayant utilisé Villo au moins 2 fois

5.2.1 SQL

```

-- Les utilisateurs ayant utilise Villo au moins 2 fois
SELECT DISTINCT trip1.user_id
3 FROM    trip trip1,
          trip trip2
WHERE trip1.user_id = trip2.user_id
AND trip1.departure_date != trip2.departure_date;

```

5.2.2 Algèbre relationnelle

$$\Pi_{trip1.user_id}(trip1 \bowtie_{trip1.user_id=trip2.user_id \wedge trip1.departure_date \neq trip2.departure_date} trip2)$$

5.2.3 Calcul relationnel

$\{t1.user_id | trip(t2) \wedge trip(t1) \wedge trip1.user_id = trip2.user_id \wedge trip1.departure_date \neq trip2.departure_date\}$

5.3 Les paires d'utilisateurs ayant fait un trajet identique

5.3.1 SQL

```
-- Les paires d'utilisateurs ayant fait un trajet identique

SELECT DISTINCT t1.user_id,
4             t2.user_id
FROM   trip t1,
      trip t2
WHERE  t1.departure_station_id = t2.departure_station_id
      AND t1.arrival_station_id = t2.arrival_station_id
9      AND t1.user_id < t2.user_id;
```

5.3.2 Algèbre relationnelle

$$\Pi_{t1.user_id, t2.user_id}(\quad) \quad (5)$$

$$t1 \bowtie_{t1.departure_station_id=t2.departure_station_id \wedge t1.arrival_station_id=t2.arrival_station_id \wedge t1.user_id < t2.user_id} t2 \quad (6)$$

5.3.3 Calcul relationnel

$$\{t1.user_id, t2.user_id | trip(t2) \wedge trip(t1) \wedge t1.departure_station_id = t2.departure_station_id \quad (7)$$

$$\wedge t1.arrival_station_id = t2.arrival_station_id \wedge t1.user_id < t2.user_id \quad (8)$$

$$\wedge trip1.departure_date \neq trip2.departure_date\} \quad (9)$$

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

5.4.1 SQL

```
1 -- Les velos ayant deux trajets consecutifs disjoints (station de retour du
-- premier trajet differente de la station de depart du suivant)

-- Optimisation possible : Pour chaque velo, trier par date de depart et regarder 2 par
2, bon apres comme faire en sql ?

6 SELECT t1.bike_id AS bike_id
   FROM trip t1, trip t2
   WHERE t1.bike_id=t2.bike_id AND
         t1.arrival_station_id!=t2.departure_station_id AND
         t1.arrival_date < t2.departure_date AND
11  NOT EXISTS (SELECT t3.departure_station_id FROM trip t3
                WHERE t1.bike_id=t3.bike_id AND
                     t1.arrival_date<t3.departure_date AND
                     t3.departure_date<t2.departure_date)
   GROUP BY t1.bike_id;
```

5.4.2 Algèbre relationnelle

5.4.3 Calcul relationnel

5.5 Les utilisateurs, la date d'inscription, le nombre total de trajet effectués, la distance totale parcourue et la distance moyenne parcourue par trajet, classés en fonction de la distance totale parcourue

```
-- Les utilisateurs, la date d'inscription, le nombre total de trajet effectués,
-- la distance totale parcourue et la distance moyenne parcourue par trajet,
-- classes en fonction de la distance totale parcourue

5 --- Barycentre des stations villos: (50.84715777660501, 4.36273021165704)
--- Perimetre Ouest-Est de la terre a cette latitude: 70.2861540037661 km/
--- Perimetre Nord-Sud de la terre: 110.94625213273459 km/
--- Distance entre 2 points (lat1,long1), (lat2,long2) ->
---      Sqrt((abs(lat2-lat1)*110)^2 + (abs(long2-long1)*70)^2)

10 SELECT      subscriber.firstname,
              subscriber.lastname,
              subscriber.entry_date,
              COUNT(trip.arrival_station_id) AS trip_count,
15              SUM(12309.070862300314*ABS(from_.latitude-to_.latitude)*ABS(from_.latitude-to_.
              latitude) + 4940.143444641126*ABS(from_.longitude-to_.longitude)*ABS(from_.longitude-
              to_.longitude)) AS km_squared
FROM          trip
INNER JOIN    station AS from_ ON trip.departure_station_id = from_.id,
              station AS to_ ON trip.arrival_station_id = to_.id,
              subscriber ON trip.user_id = subscriber.user_id
20 WHERE      trip.arrival_station_id NOT null
GROUP BY     subscriber.firstname,
              subscriber.lastname
ORDER BY     count(trip.arrival_station_id);
```

5.6 Les stations avec le nombre total de vélos déposés dans cette station (un même vélo peut-être comptabilisé plusieurs fois) et le nombre d'utilisateurs différents ayant utilisé la station et ce pour toutes les stations ayant été utilisées au moins 10 fois.

```
-- Les stations avec le nombre total de velos deposees dans cette station
2 -- (un meme velo peut-etre comptabilise plusieurs fois) et le nombre
-- d'utilisateurs differents ayant utilise la station et ce pour toutes les
-- stations ayant ete utilisees au moins 10 fois.

7
SELECT station.name, COUNT(trip.arrival_station_id), COUNT(DISTINCT trip.user_id)
FROM station, trip
WHERE trip.arrival_station_id = station.id
GROUP BY station.id
12 HAVING COUNT(trip.arrival_station_id) >= 10;
```