INFO-H303 : Base de données
Projet : Trottinettes

Groupe V : Boris COQUELET, Pierre VANDER EYKEN, Delphine SOMERHAUSEN

Langages

- Java version 12 : meilleure connaissance
- JavaFX version 11
- Apache Derby version 10.14 : OpenSource et déjà utilisée avant

Types de variable 1

- Quand cela était possible, il a été choisi d'utiliser des entiers les plus petits possibles : économie d'espace
  - Trottinette.TID = SMALLINT
  - Utilisateur.UID = INTEGER
  - Tous les mots de passe = SMALLINT
  - Tous les niveaux de charge = SMALLINT entre 1 et 4 inclus
  - Intervention.EnService = SMALLINT entre 0 et 1
  - Tous les numéros de téléphone = INTEGER
  - Tous les comptes = BIGINT
  - Trottinette.Plainte = SMALLINT

Types de variable 2

- Intervention.Note = VARCHAR(200), pouvant être null
- Trottinette.Etat = VARCHAR(20) (‘libre’, ‘en charge’, ‘occupée’, ‘défectueuse’)
- Technicien.TechID = VARCHAR(20) : trop long pour un BIGINT
- Toutes les dates = TIMESTAMP sauf Techinicien.DateEmbauche = DATE car pas d’heure
- Toutes les coordonnées de positions = DOUBLE

Méthode d'extraction

- Les fichiers en format CSV ont été modifié avant extraction :
  - Les ; ont été changés en ,
  - Les dates en format YYYY-MM-ddTHH:mm:ss ont été changés en YYYY-MM-dd HH:mm:ss
- La librairie opencsv (version 4.6) afin de parse les CSV en ArrayList, ces listes ont été parcourues et leurs éléments ajoutés à la DB
- Les fichiers XML ont été parse par balise en Arraylist, ces listes ont été parcourues et leurs éléments ajoutés à la DB

Initialisation

- Trottinette:
  - Par défaut, Etat = ‘libre’
  - PositionX et PositionY = positons du dernier trajet ou de la dernière recharge impliquant cette trottinette

- Intervention :
  - Par défaut, EnService = 1
  - Note est null à l’initialisation

Hypothèses

- Recharge:
  - Commence au moment où le bouton est appuyé
  - Finit quand le bouton est poussé de nouveau
  - DestX et DestY doivent être entrés manuellement
- Trajet :
  - Pas demandé mais possibilité de rajouter des trajets dans la DB s'il faut l'implémenter
- Trottinette :
  - Plainte s'incrémente à chaque plainte et est remis à zéro s'il y a intervention

Requête R1

- La liste et la localisation des trottinettes actuellement disponibles.
- $\pi_{TID,PositionX,PositionY}(\sigma_{Etat="libre"}(Trottinette))$
- {t.TID, t.PositionX, t.PositionY | Trottinette(t) ∧ t.Etat = 'libre' }
- SELECT TID, PositionX, PositionY

FROM Trottinette

WHERE Etat = 'libre'

Requête R2

- La liste des utilisateurs ayant utilisé toutes les trottinettes qu'ils ont rechargées.
- T ⟵ Trajet, R ⟵ Recharge, RU ⟵ Rechargeur
- TR ⟵ $\pi_{R.*}\left(T \bowtie_{T.UID=R.UID,T.TID=R.TID} R\right)$

$$\pi_{UID}(RU) - \pi_{UID}(R - TR)$$

- {ru.UID | RU(ru) ∧ ∀r (R(r) ∧ r.UID=ru.UID → ∃t (T(t) ∧ r.UID=t.UID ∧ r.TID=t.TID)}
- SELECT UID FROM RU WHERE UID NOT IN

(SELECT R.UID FROM R WHERE NOT EXISTS

(SELECT T.UID FROM T WHERE T.UID=R.UID AND T.TID=R.TID))

Requête R3

- La trottinette ayant effectué la plus grande distance depuis sa mise en service.
- SELECT TID

```
FROM TRAJET
GROUP BY TID
ORDER BY SUM((DestX-SourceX)*(DestX-SourceX) +
                (DestY-SourceY)*(DestY-Source))) DESC
FETCH FIRST ROW ONLY
```

Requête R4

- Les trottinettes ayant déjà fait l'objet d'au moins une dizaine de plaintes.
- SELECT TID

FROM INTERVENTION

GROUP BY TID

HAVING COUNT(TID) > 9

Requête R5

- Les utilisateurs ayant déjà réalisé au moins 10 trajets avec pour chaque utilisateur concerné : la durée moyenne de ses trajets en trottinette, le nombre total de trajets réalisés, le montant total dépensé en trajets.
- SELECT UID, AVG({fn timestampdiff(SQL_TSI_MINUTE, DateDepart, DateFin)}), COUNT(UID), SUM(CASE

WHEN ({fn timestampdiff(SQL_TSI_MINUTE, DateDepart, DateFin)} > 1440

THEN 1 + 36*FLOOR5({fn timestampdiff(SQL_TSI_MINUTE, DateDepart, DateFin)}/ 1440)

WHEN ({fn timestampdiff(SQL_TSI_MINUTE, DateDepart, DateFin)} > 60

THEN 1 + 6,5 * FLOOR(({fn timestampdiff(SQL_TSI_MINUTE, DateDepart, DateFin)}) / 60)

ELSE 1 + 0,15 * ({fn timestampdiff(SQL_TSI_MINUTE, DateDepart, DateFin)}

END)

FROM TRAJET GROUP BY UID HAVING COUNT(UID) >= 10