Projet BDR – Project ChooChoo

À la conquête des rails suisses : un projet pour parcourir et répertorier chaque partie du réseau ferroviaire, un tronçon à la fois.

Sacha Butty & Loïc Herman

21 janvier 2024

HEIG-VD

Sommaire

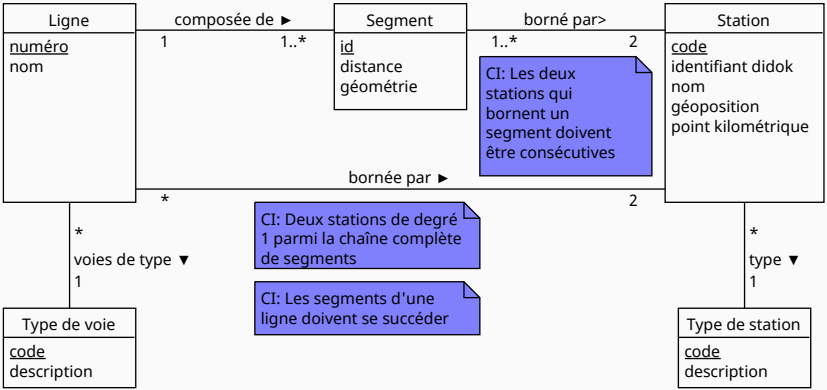
Contexte

Problématique

- **Objectif** : Développer une application pour enregistrer les tronçons de voies ferrées parcourus en Suisse
- **Données de référence** : Intégrer les stations et lignes des CFF et partenaires dans une base de données relationnelle
- **Enregistrement utilisateur** : Chaque utilisateur peut suivre ses propres parcours ferroviaires et les enregistrer dans l'application
- **Exploration systématique** : Encourager l'exploration complète du réseau ferroviaire suisse

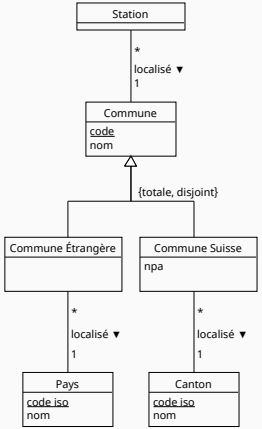
Modélisation

Choix d'implémentation – Données de référence



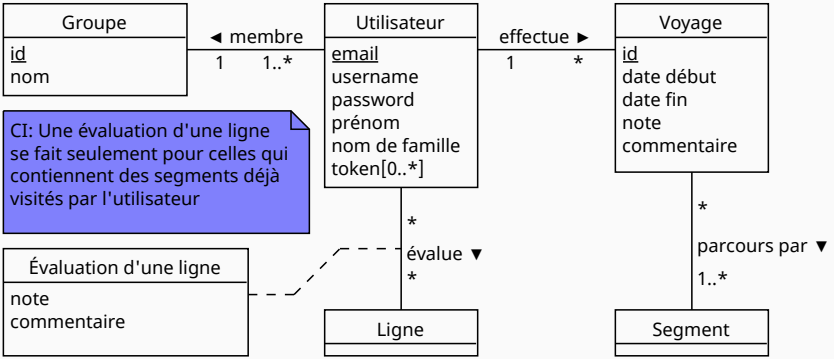
Représentation des données de références pour les lignes et stations CFF et partenaires

Choix d'implémentation – Données de référence



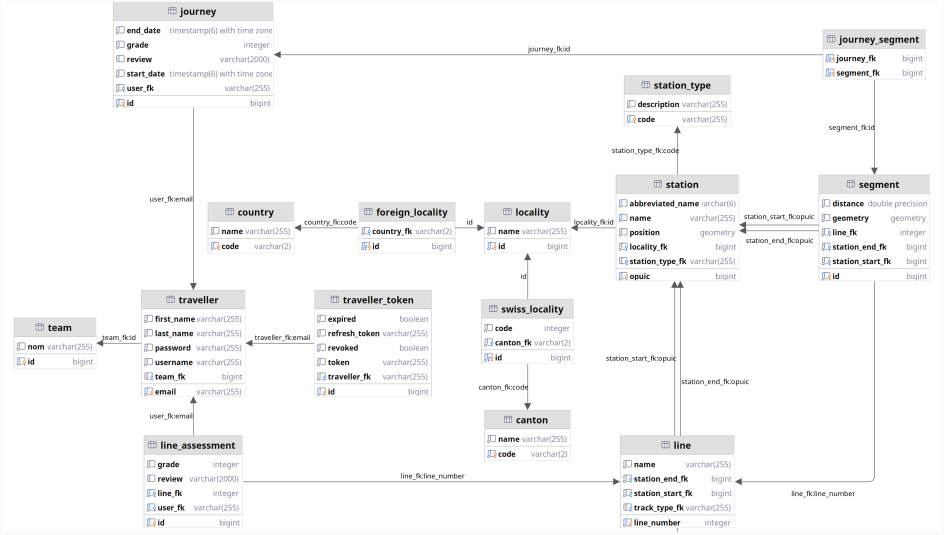
Représentation de la commune d'une station

Choix d'implémentation – Données utilisateur



Représentation des données utilisateur

Modèle relationnel



Spécificités d'implémentation

Résolution de chemins avec BFS

```
WITH RECURSIVE
    reachable(fromuid, touid, edge_id)
        AS (SELECT station.opuic, prev.station_start_fk, prev.id
            FROM station
                     JOIN segment prev ON station.opuic = prev.station_end_fk
            UNION
            SELECT station.opuic, next.station_end_fk, next.id
            FROM station
                     JOIN segment next ON station.opuic = next.station_start_fk),
    distance(uid, distance, path, edges)
        AS (SELECT :end_id::BIGINT, 0, ARRAY [:end_id]::BIGINT[], ARRAY []::BIGINT[]
            UNION ALL
            SELECT a.fromuid, b.distance + 1,
                   a.fromuid || b.path, a.edge_id || b.edges
            FROM reachable a
                     JOIN distance b ON a.touid = b.uid
            WHERE NOT (b.path @> ARRAY [a.fromuid]))
SELECT d.path, d.edges
FROM distance d
WHERE uid = :start_id
LIMIT 1;
```

Pourcentage de complétion par canton

```sql
CREATE OR REPLACE VIEW completion_by_canton AS
WITH segments_by_canton AS (
    SELECT segment.id                                    AS segment_id,
           c.code                                        AS canton_code,
           c.name                                        AS canton_name,
           COUNT(segment.id) OVER (PARTITION BY c.code) AS total_count
    FROM segment
             JOIN station s1 ON segment.station_start_fk = s1.opuic
             JOIN station s2 ON segment.station_end_fk = s2.opuic
             JOIN swiss_locality sl1 ON s1.locality_fk = sl1.id
             JOIN swiss_locality sl2
                  ON s2.locality_fk = sl2.id AND sl1.canton_fk = sl2.canton_fk
             JOIN canton c ON sl1.canton_fk = c.code
)
SELECT t.email                       AS user_email,
       t.team_fk                     AS team_fk,
       fs.canton_code                AS canton_code,
       fs.canton_name                AS canton_name,
       COUNT(DISTINCT js.segment_fk) AS travelled_count,
       fs.total_count                AS total_count
FROM segments_by_canton fs
         CROSS JOIN traveller t
         LEFT JOIN journey j ON t.email = j.user_fk
         LEFT JOIN journey_segment js ON j.id = js.journey_fk AND js.segment_fk = fs.segment_id
GROUP BY fs.canton_code, fs.canton_name, fs.total_count, t.email;
```

Démonstration