Haute école du paysage, d'ingénierie et d'architecture

Systèmes de bases de données

PokeDB

B. Coudray, Q. Berthet, K.Torres

2019-2020

Table des matières

1 Présentation du projet

Dans le cadre du cours de systèmes de base de données, nous avons décidé de réaliser une base de données Pokémon comme projet du cours. Ce sujet nous intéresse et avec plus de 20 ans d'existence, la licence Pokémon a eu le temps de se développer nous offrant ainsi un vaste sujet d'étude. Finalement, nous avons aussi choisi ce sujet par commodité car, toute personne née après les années 90 à au moins eu une fois une expérience avec l'univers Pokémon, que ce soit via les jeux, les séries, voir même pour les plus adeptes, les cartes.

2 Modèle entité-association

2.1 Version intermédiaire

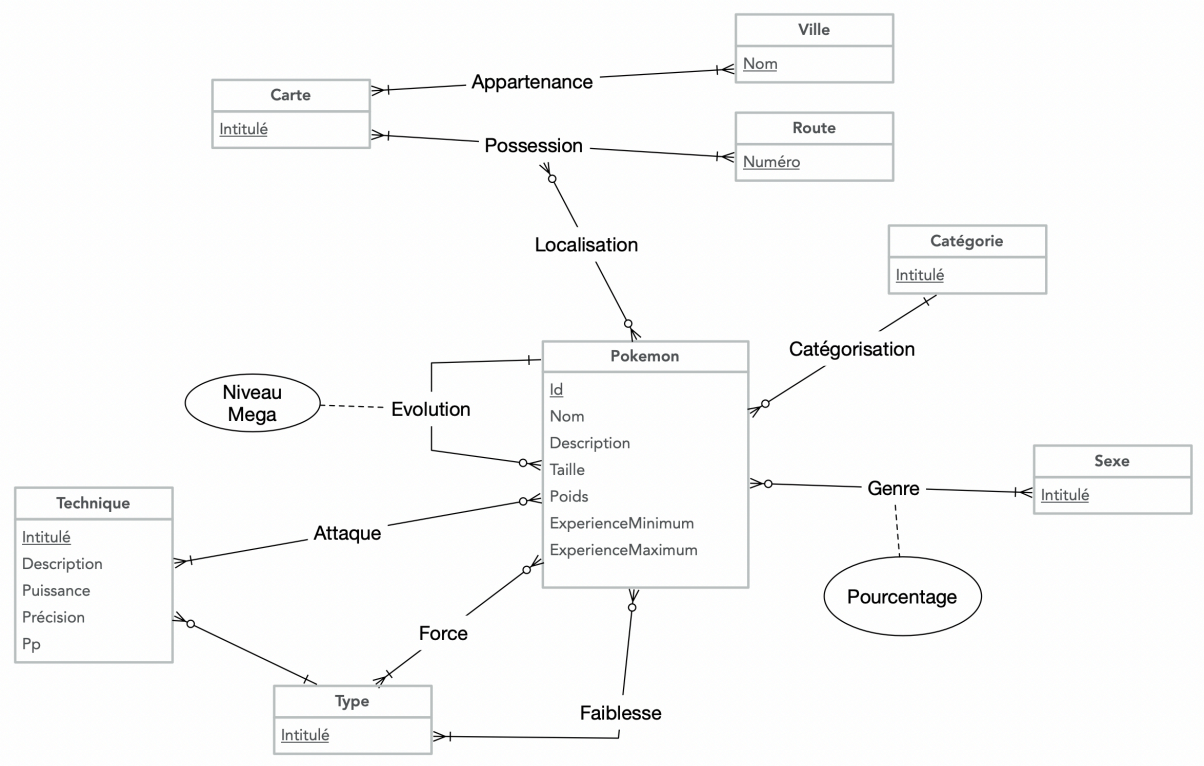

FIGURE 1 – Modèle entité-association intermédiaire

Dans cette version intermédiaire de notre modèle EA, plusieurs erreurs nous ont été remontées. Parmi elles, la non-présence du type d'association entre l'entité "Technique" et "Type" et la mauvaise notation de l'association ternaire entre "Carte", "Route" et "Pokemon". En effet, nous n'avions pas encore vu la théorie sur les associations entre plusieurs entités, nous avions donc improvisé la notation en attendant la leçon sur ce sujet.

2.2 Version finale

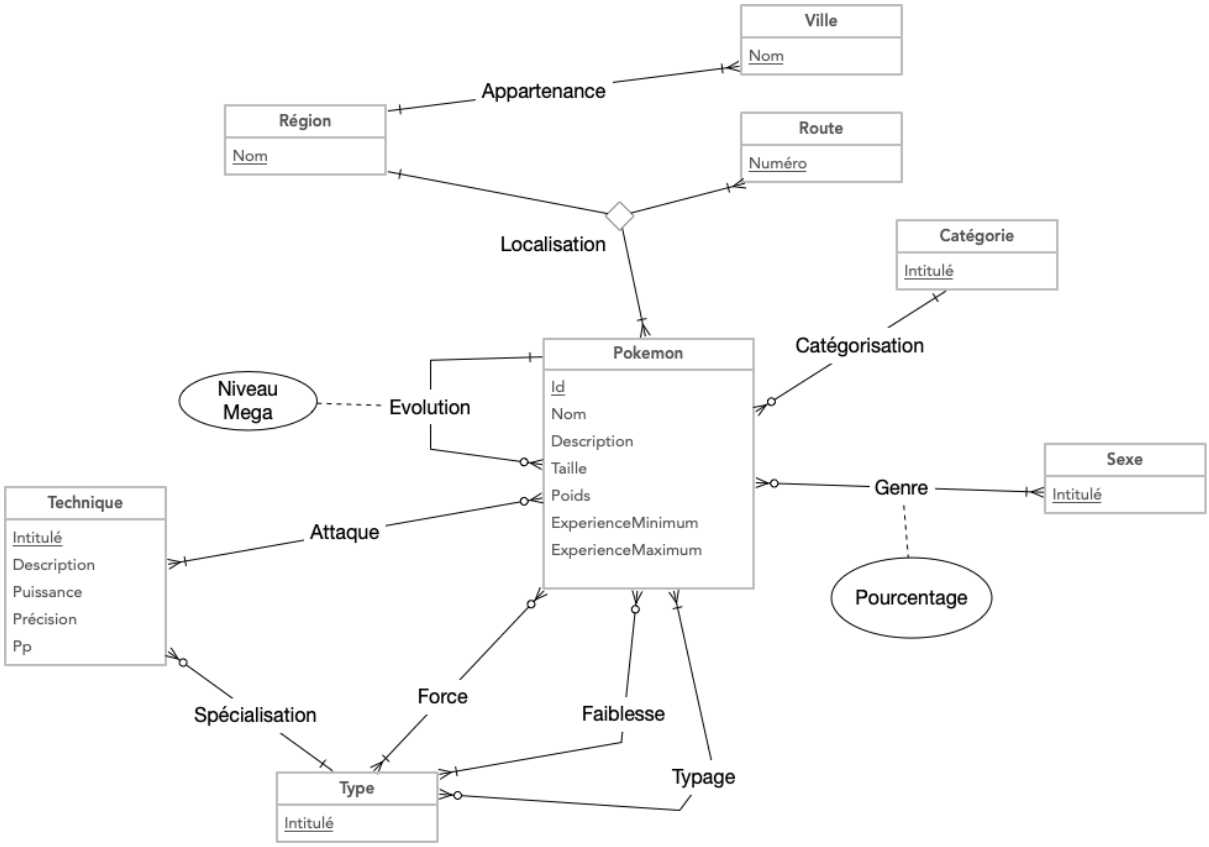

FIGURE 2 – Modèle entité-association final

Dans la version finale de notre modèle EA, nous avons corrigé les deux erreurs énoncées précédemment. Comme on peut le voir sur l'image, le type d'association entre "Technique" et "Type" est "Spécialisation". En effet, une technique est spécialisée dans un seul type possible (par exemple, la technique "Danseflamme" est de type feu). De plus, nous avons corrigé l'association ternaire suite à la leçon sur la modélisation avancée. La localisation d'un Pokémon est définie selon les dépendances fonctionnelles suivantes :

- connaissant une région et une route, il y a un ou plusieurs Pokémon qui se trouvent à cet emplacement,
- connaissant une route et un Pokémon, il y a qu'une seule région,
- connaissant une région et un Pokémon, il peut se trouver sur une ou plusieurs routes de la région.

On peut noter le changement de cardinalité entre l'entité "Région" et "Route" (plusieurs à plusieurs vers un à plusieurs) ainsi qu'entre "Région" et "Ville" (plusieurs à plusieurs vers un à plusieurs). Après quelques recherches, nous avons remarqué que chaque numéro de route était unique entre chaque région et qu'une ville n'était présente que dans une région.

Une association récursive définie sur l'entité "Pokémon" permet de représenter les évolutions d'un Pokémon. Ainsi, celui-ci peut avoir zéro ou plusieurs évolutions et à l'inverse, un Pokémon a évolué depuis un Pokémon. La triple association entre l'entité "Pokémon" et "Type" permet de représenter les forces, les faiblesses et le typage d'un Pokémon, celui-ci peut en avoir un ou plusieurs. Par exemple, les types de forces de Dracaufeu sont les types glace et plante et ses faiblesses sont les types roche et eau.

2.3 Forces

Nous pouvons lister les forces suivantes pour notre modèle EA :

— l'association ternaire entre "Région", "Route" et "Pokémon" permet de représenter à la fois la localisation d'un Pokémon mais aussi les routes disponibles dans une région.

— l'association récursive pour représenter les évolutions est élégante, elle évite d'avoir une autre entité similiare à Pokémon pour enregistrer les évolutions.

2.4 Faiblesses

Nous pouvons lister les faiblesses suivantes pour notre modèle EA :

— un Pokémon peut avoir une force qui est aussi une faiblesse, c'est une contrainte statique non modélisable

— un Pokémon peut posséder une attaque alors que celle-ci est du même type qu'une faiblesse, c'est une contrainte statique non modélisable

3 Modèle logique de données

3.1 Version intermédiaire

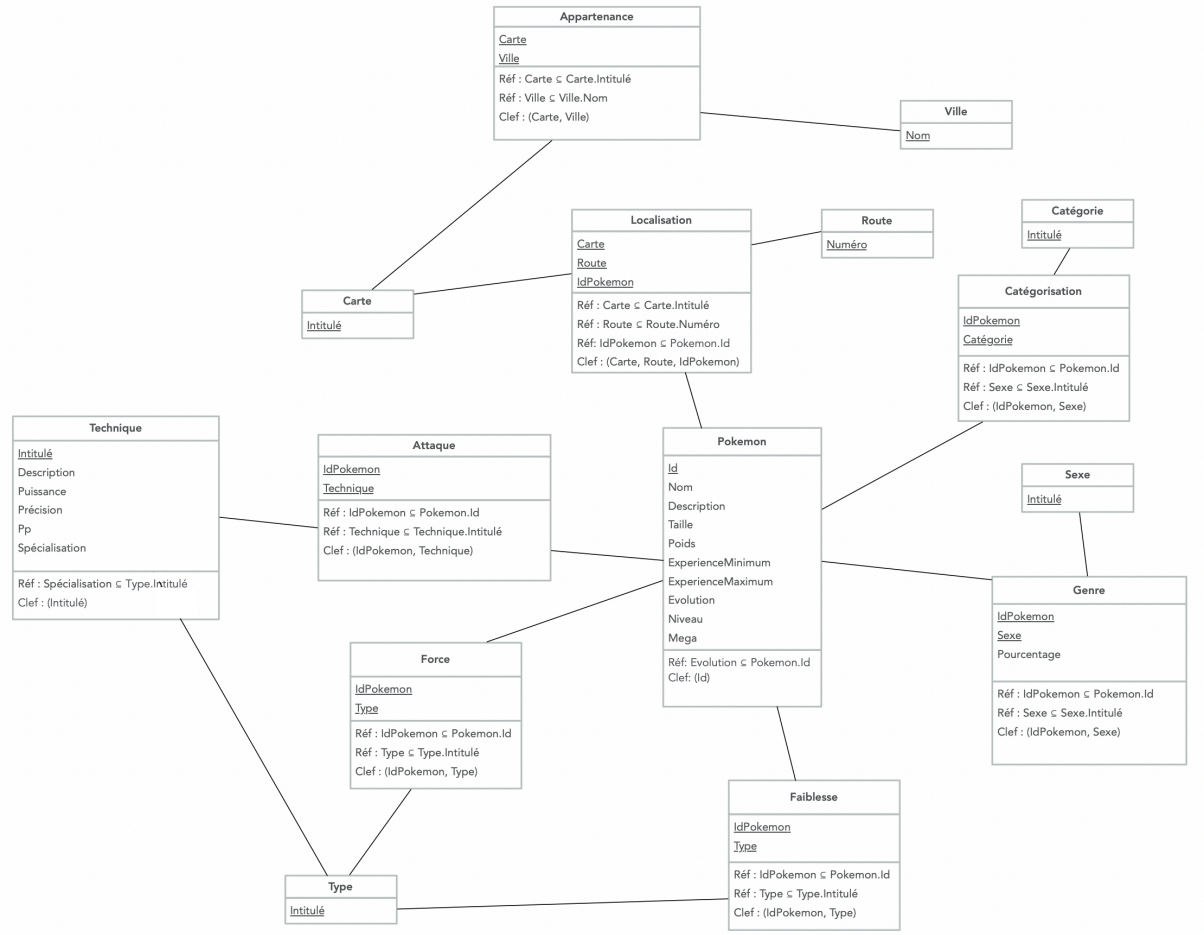

FIGURE 3 – Modèle logique de données intermédiaire

La transformation du modèle EA vers le modèle logique de données met en avant le choix des clefs primaires restrictifs sur l'ensemble des types d'association (plusieurs à plusieurs) devenus table. On peut noter l'utilisation d'une clef primaire utilisant les 3 clefs étrangères dans l'association ternaire. Celle-ci permet par exemple d'avoir :

— plusieurs fois la même route sur plusieurs cartes,
— plusieurs Pokémon sur une même route
— plusieurs Pokemon dans une carte

3.2 Version finale

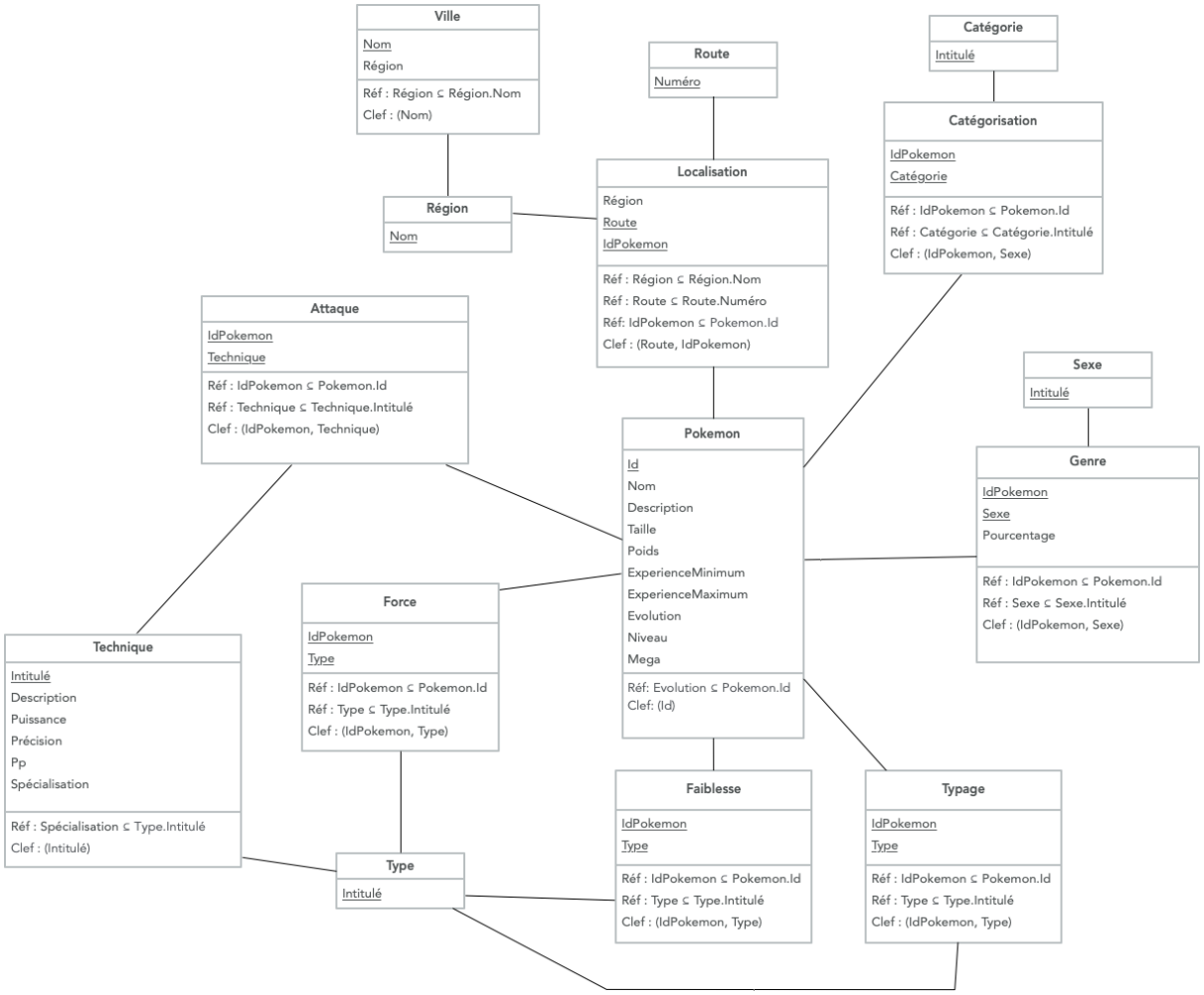

FIGURE 4 – Modèle logique de données final

La version finale ne change que très peu par rapport à la version intermédiaire présentée précédemment. En effet, seule la clef primaire de la table "Localisation" a été modifiée pour respecter les dépendances fonctionnelles du modèle EA finale. Cependant, on peut remarquer que la table "Carte" a été renommé en "Région" pour éviter toute ambiguïté et une nouvelle table "Typage" est apparu car lors de la création des requêtes, on s'est rendu compte qu'on ne pouvait pas savoir le type d'un Pokémon.

3.3 Forces

Nous pouvons lister les forces suivantes pour notre modèle logique de données :

- utilisation massive de clefs primaires restrictives, les métiers permettent d'en abuser à bon escient :
  - un Pokémon ne peut pas avoir plusieurs fois la même technique, force, faiblesse, typage, sexe et catégorie.

3.4 Faiblesses

Nous pouvons lister les faiblesses suivantes pour notre modèle logique de données :
— un Pokémon peut avoir une force qui est aussi une faiblesse, c'est une contrainte statique non modélisable,
— un Pokémon peut posséder une attaque alors que celle-ci est du même type qu'une faiblesse, c'est une contrainte statique non modélisable

4 Requêtes intéressantes en algèbre relationnelle

1. Lister les attributs et les routes où on peut trouver le Pokémon avec l'id 1.
   $R \leftarrow Localisation \bowtie_{Localisation.IdPokemon=Pokemon.Id} Pokemon$
   $\pi_{\{Route,Region,Nom,Description,Taille,Poids\}}(\sigma_{Pokemon.Id=1}(R))$

2. Lister tous les Pokémon qui ont une evolution Mega.
   $\pi_{\{Id,Nom\}}(\sigma_{Pokemon.Mega=True}(Pokemon))$

3. Lister tous les Pokémon qui n'ont pas de forces.
   $\pi_{\{Id,Nom\}}Pokemon - \pi_{\{Id,Nom\}}(Pokemon \bowtie_{Pokemon.Id=Force.Id} Force)$

4. Lister les categories des Pokémon trouvés dans la route 101.
   $R \leftarrow Categorie \bowtie_{Categorie.IdPokemon=Pokemon.Id} Pokemon \bowtie_{Pokemon.Id=Localisation.IdPokemon} Route$
   $\pi_{\{IdPokemon,Categorie\}}(\sigma_{Localisation.Route=101}(R))$

5. Lister les Pokémon faibles contre le type "Eau" se trouvant dans la route 102.
   $R1 \leftarrow \sigma_{Faiblesse.Type="Eau"}(Pokemon \bowtie_{Pokemon.Id=Faiblesse.IdPokemon} Faiblesse)$
   $R2 \leftarrow \sigma_{Localisation.Route=102}(R1 \bowtie_{R1.Id=Localisation.IdPokemon} Localisation)$
   $\pi_{\{Id,Nom\}}(R2)$

6. Lister tous les Pokémon de catégorie "Graine" et de type "Plante" qui ont une faiblesse contre le "Feu".
   R1 $\leftarrow$ (Pokemon $\bowtie_{\text{Pokemon.Id = Catégorisation.IdPokemon}}$ ($\sigma_{\text{Catégorisation.Catégorie = "Graine"}}$(Catégorie))
   R2 $\leftarrow$ (R1 $\bowtie_{\text{R1.Id = Typage.IdPokemon}}$ ($\sigma_{\text{Typage.Type = "Plante"}}$(Type))
   R3 $\leftarrow$ (R2 $\bowtie_{\text{R2.Id = Faiblesse.IdPokemon}}$ ($\sigma_{\text{Faiblesse.Type = "Feu"}}$(Type))
   $\pi_{\{\text{Id, Nom}\}}$(R3)

7. Lister tous les Pokémon de la route 103 qui n'ont pas d'évolution.
   $R1 \leftarrow \pi_{(Id,Nom)}(\sigma_{Pokemon.Evolution="NULL"}(Pokemon))$
   $R2 \leftarrow \pi_{(Id,Nom)}(Pokemon \bowtie_{Pokemon.Id=Localisation.IdPokemon} Localisation)$
   $\pi_{\{Nom\}}\sigma_{Localisation.Route=103}(R1 \bowtie_{R1.Id=R2.Id} R2)$

8. Lister toutes les villes de la région "Kanto".
   $\pi_{\{Nom\}}(\sigma_{Region="Kanto"}(Ville))$

9. Lister tous les Pokémon qui sont localisés dans toutes les routes.
   $R \leftarrow (\text{Pokemon} \bowtie_{\text{Pokemon.Id = Localisation.IdPokemon}} (\textit{Localisation})$
   $\pi_{\{\text{Id, Numéro}\}}(R) \ / \ \pi_{\{\text{Numéro}\}}(\text{Route})$

10. Lister tous les Pokémon femelles ayant comme attaque "Morsure" et se trouvant sur la route 1.
    $R1 \leftarrow (\text{Pokemon} \bowtie_{\text{Pokemon.Id = Genre.PokemonId}} (\sigma_{\text{Genre.Sexe = "Femelle"}}(\text{Genre}))$
    $R2 \leftarrow (R1 \bowtie_{\text{R1.Id = Attaque.PokemonId}} (\sigma_{\text{Attaque.Technique = "Morsure"}}(\text{Attaque}))$
    $R3 \leftarrow (R2 \bowtie_{\text{R2.Id = Localisation.Id}} (\sigma_{\text{Localisation.Route = 1}}(\text{Localisation}))$
    $\pi_{\{\text{Id, Nom, Sexe, Pourcentage, Route}\}}(R3)$

5 Vues

Pour faciliter la visualisation des Pokémon et simuler un Pokédex amélioré, nous avons créé plusieurs vues :

— une vue pour chaque région (Hoenn, Johto, Kalos, Sinnoh et Unys) permettant de voir les Pokémon disponibles dans les régions respectives
— une vue affichant les Pokémon ayant une méga-évolution
— une vue affichant les Pokémon ayant pleinement évolués

6 Gestion de groupes

En nous basant sur nos vues créées, on a décidé de créer un rôle pour chaque vue et deux utilisateurs appartenant à ces rôles ont accès en lecture à la vue. Ainsi, nous avons par exemple :

— le rôle "HoennFan" a accès à la vue "Hoenn_Only",
— le rôle "JohtoFan" a accès à la vue "Johto_Only"

De plus, un rôle "PokeMaster" est disponible pour pouvoir administrer la base de données, deux utilisateurs ont ce rôle. Finalement, un rôle "App" a été créé pour les applications utilisant la base données (notamment notre Pokémon CLI). Ce rôle a accès à la base de données en lecture et contient déjà un utilisateur.

7 Procédures stockées

Nos procédures stockées ont comme objectif de faciliter les insertions et la mise à jour de la base de données. En effet, celles-ci permettent notamment de :

— définir la catégorisation d'un Pokémon
— définir le typage d'un Pokémon
— associer une attaque à un Pokémon
— associer une force à un Pokémon
— associer une faiblesse à un Pokémon
— définir le genre du Pokémon
— définir l'évolution d'un Pokémon

8 Triggers

Dans le but des respecter les contraintes statiques non modélisables, nous avons créé des *triggers* pour vérifier la cohérence des données à l'insertion. Ainsi, quatre *triggers* sont disponibles :

- avant insertion dans la table "Faiblesse", un *trigger* vérifie que celle-ci n'est pas une force ou le type du Pokémon
- avant insertion dans la table "Force", un *trigger* vérifie que celle-ci n'est pas une faiblesse ou le type du Pokémon
- avant insertion dans la table "Typage", un *trigger* vérifie que celui-ci n'est pas une faiblesse du Pokémon
- avant insertion dans la table "Attaque", un *trigger* vérifie que le type de l'attaque n'est pas une faiblesse du Pokémon

9 Distribution du travail dans le groupe

De façon à avoir un travail vraiment équilibré, on a toujours choisi de le distribuer entre nous d'une façon équivalente. Au début de ce projet, on a choisi de distribuer le travail en trois parties, le modèle entité-association, le modèle logique et les requêtes en algèbre relationnelle. Les forces et faiblesses des modèles ont été un travail commun. Pour la deuxième partie, on a continué notre façon de diviser le travail, une personne a pris en charge les procédures stockées et les *triggers*, une autre de peupler la base de données et la dernière des vues et de la gestion des droits. La création des tables s'était distribuée entre tous d'une façon équitable.

Table des figures