Base de données Sim’SSE Editor

Ce document regroupe l’étude de faisabilité et le proof of concept qui ont mené à choisir cette base de données relationelle.

Table des matières

[Object 2](#_Toc185522192)

[Modélisation Graphique 2](#_Toc185522193)

[Modélisation relationnelle 5](#_Toc185522194)

[I. Modélisation l’évolution des variables physiologiques des cas cliniques 5](#_Toc185522195)

[Dans l’application web Sim’SSE Editor 5](#_Toc185522196)

[Dans l’application mobile Sim’SSE 6](#_Toc185522197)

[II. Modéliser le principe de « template » 6](#_Toc185522198)

[Modèle 6](#_Toc185522199)

[Plastron 6](#_Toc185522200)

# Modélisation relationnelle

Notre objectif ici est d’expliquer comment nous modélisons dans la base de données relationnelle. Le modèle proposé ci-dessous est la première version du modèle physique des données.

## Modélisation l’évolution des variables physiologiques des cas cliniques

### Dans l’application web Sim’SSE Editor

Le plus gros défi dans cette partie est de stocker les informations nécessaires pour reproduire le graphique d’un cas clinique.

* Les actions sont stockées dans la table event. Ils peuvent soit être de type « Start », « Action » ou « Bioevent ».
  + Dans le premier cas, les éléments de type « Start » permettent de récupérer tous les éléments à déclencher à t0.
  + Dans le deuxième cas, on relie à l’event à l’action correspondante dans la table action qui est la table de référence des actions (modifiable dans règles).
  + Dans le troisième cas, ils sont alors reliés à la table bioevent qui sert de table de référence.
* Les tendances sont stockées dans la table trend. Elles sont reliées à une table variaphysiodefaut qui stocke les informations par défaut remplies dans les règles.

**Construction du graphique :**

* Chaque profil est relié à des variables physiologiques qui sont propre à leur profil.
* On stocke les positions de chaque nœud dans le graphique avec les attributs x,y dans les tables trend et event.
* Pour conserver les liens entre les nodes avec les flèches start, stop, (pause), les liens sont stockés dans une table qui lie un event à une nouvelle tendance, lien déclenche.
* Les timer peuvent être inclus dans la table event en tant que nouveau type, ou bien comme attribut dans le lien entre un event et une tendance pour signifier que l’effet est retardé (à trancher).

**Ajouter un modèle dans un autre modèle (principe de template pour les modèles)**

* Les symptômes sont stockés dans la table modèle et sont identifiables par un booléen symptôme.
* On cherche à reproduire le graphique qui imbrique un modèle dans un autre. On cherche ici à ajouter le modèle B dans le modèle A.

On propose de relier la table event à une table instance modèle. Cette dernière stocke l’id du modèle B ainsi que la position de la box du modèle B dans le graphique du modèle A et le type de lien entre l’event et cette dernière.

Pour permettre de modifier les caractéristiques du modèle B uniquement dans le modèle A et pas dans les autres modèles, on duplique les données du modèle B en créant un modèle C. Ainsi dans la table instance\_modele, on relie plutôt A et C au lieu de A et B.

Remarque : il existe alors une instance du modèle/symptôme à chaque fois qu’il est utilisé dans un autre modèle donc on multiplie les données même si elles sont potentiellement identiques. Pour optimiser, on pourrait imaginer que l’étape de duplication ne se fait pas automatiquement mais uniquement si l’utilisateur modifie les données dans le symptôme.

Remarque 2 : On exclue le cas où un modèle est relié à un event ou une autre trend, car cela n’a pas d’intérêt dans la modélisation et ne représente rien de réaliste.

### Dans l’application mobile Sim’SSE

Dans l’application mobile, on récupère pour chaque plastron :

* Les valeurs à t0 de chaque valeur physiologique ainsi que leur tendance.

Pour cela, on récupère le seul event de type « start » relié à notre modèle puis par les deux tables des liens les tendances et les modèles associés. Dans le cas d’un modèle associé, on fonctionne de manière récursive pour récupérer également ses tendances à t0.

* La liste des actions qui ont un impact sur les tendances du plastron avec la nouvelle tendance de la variable physiologique ainsi que la valeur de la cible.

Pour chaque event qui comprend id du modèle, on récupère toutes les actions associées via l’id action. Puis via le lien\_declenche, on récupère toutes les valeurs de tendances et les cibles. Enfin, pour chaque tendance, on récupère la variable physiologique concernée avec l’id\_varia.

* Si des modèles/symptômes sont utilisés dans le modèle, on doit également récupérer leur contenu et l’inclure dans les tendances à t0 ou résultantes des actions.
* Pour un profil particulier, on a des variables physiologiques de départ, min et max particulière, il faudra donc les récupérer également avec le lien\_varia.

## Modéliser le principe de « template »

### Modèle

On a vu dans le point précédent comme nous allions gérer les symptômes et la combinaison de modèle entre eux.

Ici, nous traitons plutôt de la validation d’un modèle, c’est-à-dire quand on définit le modèle comme terminé et qu’il peut être utilisé dans les modèles de plastron.

Ainsi, on utilise l’attribut statut sous forme de boolean, pour uniformiser le modèle.

On n’utilise donc plus d’attribut « template » dans le modèle.

### Plastron

Les plastrons peuvent être noté comme terminé, en cours ou à faire grâce à l’attribut statut.

Pour rendre possible la modification du modèle dans le plastron, sans modifier le modèle de départ, on crée un nouveau modèle relié au plastron. De plus, si l’utilisateur veut pouvoir enregistrer son modèle de plastron en tant que nouveau modèle, on change l’attribut statut du modèle pour qu’il devient un modèle « Terminé » et donc un modèle utilisable pour d’autres plastrons.

Une image contenant texte, diagramme, Plan, noir et blanc

Description générée automatiquement