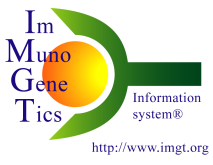
Ontologies et système d'information

Module MED *Bioinformatique et ontologies*, Université Montpellier 2

Patrice DUROUX

<Patrice.Duroux@igh.cnrs.fr>

IMGT, Institut de Génétique Humaine, UPR 1142 CNRS



24 mai 2013

À la carte

Système d'information et base de données

Le cas d'IMGT

Algèbre relationnelle

Algèbre d'intervalles

Logiques de description

Interroger le Web

Système d'information et base de données

Système (en très bref)

Un *système* est un ensemble d'éléments (matériels ou pas) «organisés» ou en «interaction» entre eux. Il comporte généralement des éléments en entrée et des éléments en sortie et où des objets (des biens ou de l'information) sont créés, détruits, échangés, reproduits, transformés, protégés... Pour nous, les systèmes seront constitués par des organisations (institution, entreprise, administration, communauté, collectif, etc.) et fonctionnent en vue de la réalisation d'objectifs déterminés (ou plus que moins).

Exemples

Exemple 1 Une entreprise qui fabrique/commercialise des produits à :

- en entrée : des produits achetés, des commandes, les paiements des clients ;
- en sortie : des produits vendus, des factures, les paiements aux fournisseurs.

Exemple 2 Un système algébrique d'équations.

Systèmes parmi les systèmes

On peut distinguer des systèmes les sous-systèmes connexes suivants :

- *système de pilotage* : régulation, contrôle, décision, définition d'objectifs ;
- *système opérationnel* (ou opérant) : réalisation d'actions ;
- *système d'information* (SI) : interface entre les deux précédents.

Système d'information

Composé d'éléments divers (personnels, machines, conventions/protocoles, ...), le SI informe le système de pilotage sur les fonctionnement du système opérant, et renvoie au système opérant des directives provenant du système de pilotage.
Le SI est la «mémoire» de l'organisation avec 2 aspects :

- statique : enregistrer des faits, des règles et des contraintes.
- dynamique : mettre à jour ses enregistrements.

Nature de l'information

Le SI d'une organisation regroupe tout ce qui, à quelque niveau que se soit, traite ou stocke ses informations relatives :

- aux flux : produits en stocks, produits commandés, bons de livraison, factures, bons de commandes...
- à l'univers extérieur : clients, fournisseurs...
- à l'organisation de l'entreprise : que se passe-t-il entre l'enregistrement d'une commande et sa livraison ?
- aux contraintes légales : lois, règlements, paramètres financiers...
- *etc.*

Banques de données

Qu'est-ce qu'une banque de données ?

- Sans informatique :
  - Elle est constituée d'une collection de fichiers et de liens logiques entre ces fichiers.
  - Un fichier est une collection de fiches.
  - Une fiche comporte un ou plusieurs renseignements sur un élément (objet ou personne) de la BD.
  - Chacun de ces renseignements (appelés aussi champs) est une information indivisible.

Exemple : les fichiers tiroirs ou casiers des bibliothèques.

Essor informatique

Fondement dans les interactions de la mathématique, de la logique et l'ingénierie des machines.
4 générations :

- 1946-1955 : tubes à vide, cathodique, tores de ferrite
- 1956-1963 : transistors
- 1964-1971 : circuits intégrés et «mini»-ordinateur
- 1971 à aujourd'hui : microprocesseur, micro-ordinateur, super-calculateur

Parallèlement : premiers travaux sur la communication entre ordinateurs vers 1961 et Internet depuis 1990

Systèmes de bases de données

- Avec informatique :
  - La définition d'une BD est identique.
  - Les concepts utilisés en informatique sont quelque peu différents.
  - Le modèle de représentation des données est considéré.

  Exemple : avec le modèle relationnel, une BD est définie par un ensemble de relations (des tables et des dépendances entre ces tables).

Historiquement (les années 60)

- Tout d'abord, l'apparition des premiers systèmes d'exploitation (SE) : faciliter la gestion physique de l'«information» (données/programmes) en mémoire (mécanique, électronique, ...) et de son stockage (papier, magnétique, ...)
- Puis les systèmes de gestion de fichiers (SGF) : séquentiel, relatif, indexé, séquentiel indexé, ... => indépendance «relative» des programmes et des données.
- Ensuite, les systèmes de gestion de bases de données (SGBD) : privilégier la facilité de consultation, des logiciels plus complexes pour la création et la mise à jour.

Apparition des SGBD

- Conférence «Development and Management of a computer-centered data base» (Santa Monica, 1964).
- Considération pour de gros volumes d'informations avec souci d'efficacité des accès.
- SGBD est un système de stockage des données et assure une facilité de maintenance.

Base $\neq$ Banque = index ou ensemble d'index (le référentiel versus le factuel).

Évolution du contexte informatique

- Matériel informatique : temps d'accès et de calcul $\searrow$, capacité mémoire $\nearrow$, coût $\searrow$.
- Systèmes et logiciels : utilisateurs multiples, systèmes distants.
- Interconnexion et réseau : communication et disponibilité des réseaux.
- Développement d'applications : génie logiciel, environnement intégré, modèle 4 Génération (MVC).
- Infrastructure : poste-à-poste, grand calculateur, grille, «clouds»...

Qu'est-ce qui peut être informatisé ?

D'une manière générale, seule une partie du SI peut-être automatisée, c'est ce que l'on appelle *système automatisé d'information* (SAI).
Le SAI communique avec son environnement extérieur par des saisies et des accès. Il contient une partie mémorisation et une partie traitement.

- Mémorisation : stockage des données, structuration des données, règles de fonctionnement.
- Traitement : contrôle des données, mise à jour, recherches, calculs.
- Interface entre l'extérieur (univers) et le SAI : saisie, accès.

Concevoir un SI

De nos jours l'informatique est tant une technologie qu'une discipline pour les SI.
L'analyse (informatique) d'un SI aboutie traditionnellement aux modèles :

- de données (MDD), la formalisation des données (informations de toute nature) présentes à un moment ou un autre dans le système ;
- de traitement (MDT), la formalisation des traitements (i.e des processus dynamiques) intervenant dans le système.

Approche par couches

8 métier
couvrir l'ensemble des problématiques liées à l'exécution des tâches liées au métier que le système d'information est censé outiller (acteurs, scénarios, objets métiers)

7 applicative ou processus
donner la composition des objets métiers selon les informations (numériques), les représentations, leur localisation. (cas d'usage)

6 architecture système
établir les éléments logiciels à assembler pour produire et mettre en scène les objets et procédures métier (urbanisation, protocoles, services, serveurs)

Approche par couches

5 ergonomie
déterminer des «classes» d'information dans leur contexte procédural afin de définir une sémiotique et une hiérarchie d'affichage adéquate au regard des couches inférieures (interface, ergonomie, layout, widgets, sémiotique)

4 architecture logicielle spécifier des développements logiciels propres à l'aide (ou non) des outils de l'ingénieurie logicielle (composants, patterns, middleware, façades)

3 algorithme (ou programmation)
développer les logiciels spécifiques dans les langages de programmation (variables, types, fonctions, boucles, structures conditionnelles)

Approche par couches

2 système
gèrer l'accès aux ressources mémoire, fichiers et pseudo-fichiers, les droits et les autorisations, les processus (librairies et commandes système,système de fichiers,processus et IPC)

1 microprogrammation
envisager des extensions en langage «bas niveau» pour le support aux langages de «haut niveau» (modules,micro-systèmes)

0 matériel
adapter aux organes périphériques (pilotes, micro-codes, logiciel embarqué)

Périmètre du formalisé

- Données formalisées : renseignements sur un client, une commande, les cours de la bourse, la température et pression.
- Données non formalisées : conjoncture économique, météo, rendement d'un individu (frontière floue).
- Traitements formalisés : édition d'un facture, renouvellement des stocks, lettre de rappel.
- Traitements non formalisés : si le fournisseur ne donne pas satisfaction alors en changer, arrêter la fabrication d'un produit et en créer un autre.

Difficultés

Le problème du traitement des choix :

- rarement automatisables, alors du ressort de l'être humain ;
- sauf cas particuliers, il est possible de les formaliser dans le modèle.

Exemple : le renouvellement d'un stock sera du ressort de l'être humain, sauf si on peut le formaliser par un règle comme «si stock(xx)<..., alors commander(xx,yy)».

Outils de modélisation

MERISE : Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise

- Fruit d'une consultation nationale du Ministère (fin années 70).
- Méthodologie : modèles conceptuel, logique et physique.
- Approche données/traitements.

UML : Unified Modeling Language

- Factorisation d'approche existantes : OMT, Booch et OOSE (années 90).
- Notation (type de diagrammes) et recommandation.
- Approche dite «objets».

Evolution

Le système est dynamique, il est fait pour bouger :

- Ajout (extension), retrait (restriction) ou remplacement (départ, panne) d'éléments,
- Performance, efficacité, optimisation
- Traçabilité des objets métiers,
- Sécurisation : cohérence, fiabilité, violation de droit et responsabilité.

Qu'elle approche pour

- Faciliter le concensus et la cohérence,
- Spécifier les objets en entrée et à la sortie,
- Armoniser le vocabulaire et les définitions,
- Se guider et rechercher dans le système (fouille de données).

=> Ontologie

Le cas d'IMGT

Infrastructure

Répartie :

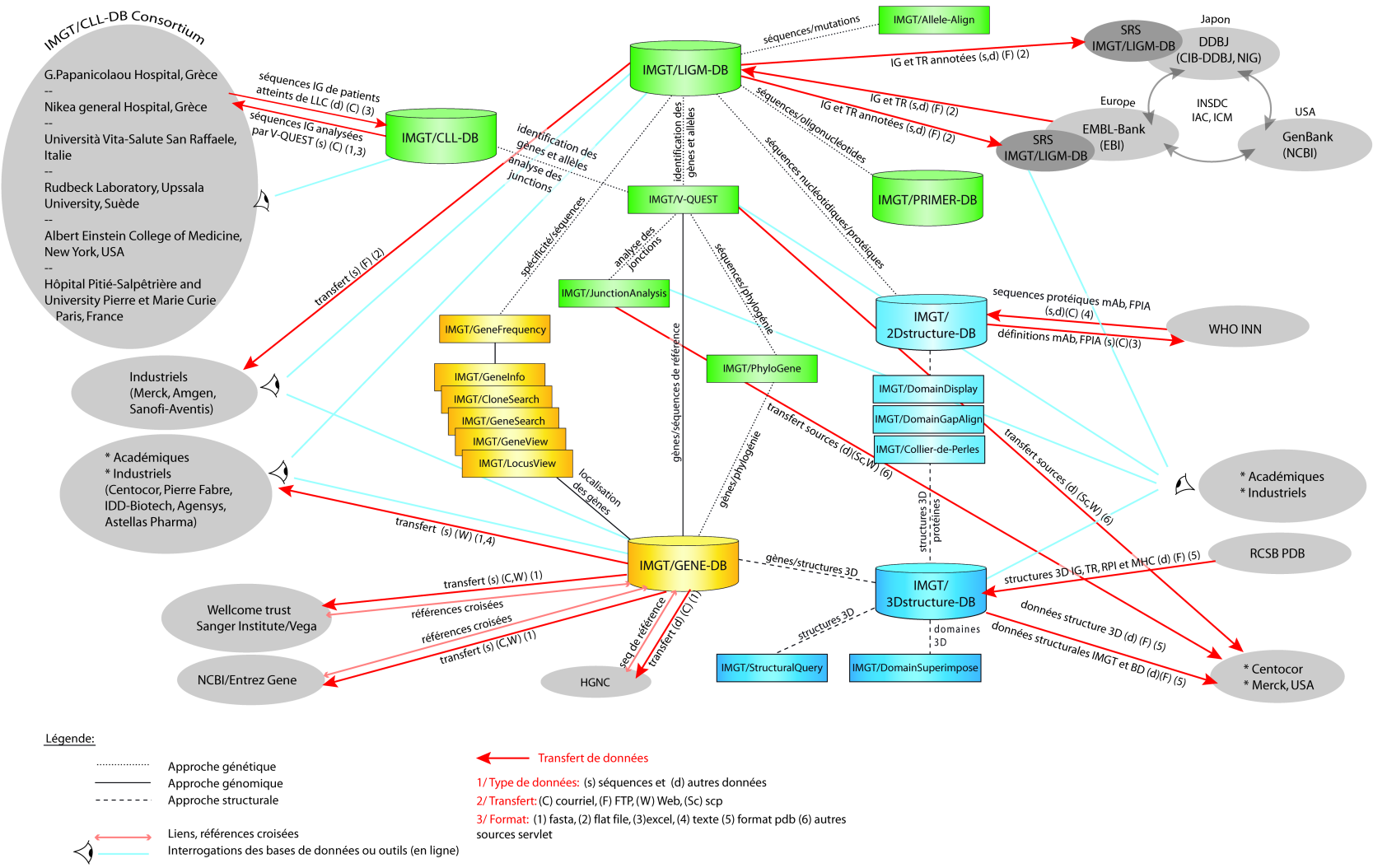
- personnel sur 2 sites (IGH et UM2) : management, informaticiens, bio-informaticiens, bio-curateurs.
- informatique sur 3 sites (IGH, UM2 et CINES) : 10 serveurs + 15 postes PC.

Hétérogène :

- compétence pluri-disciplinaire
- systèmes informatiques variés : SE (Windows, Linux), SGBD (Sybase, MySQL), langages de programmation (Shell, Perl, Java, PHP)
- formats bio-informatiques : texte «brut» ou tabulée (CSV), données séquences et structures (FASTA, EMBL, PDB), documents (HTML, PDF), diagrammes (bitmap, SVG)

Elements logiciels et leur relations

Relations entre IMGT® et ses différents collaborateurs et utilisateurs.

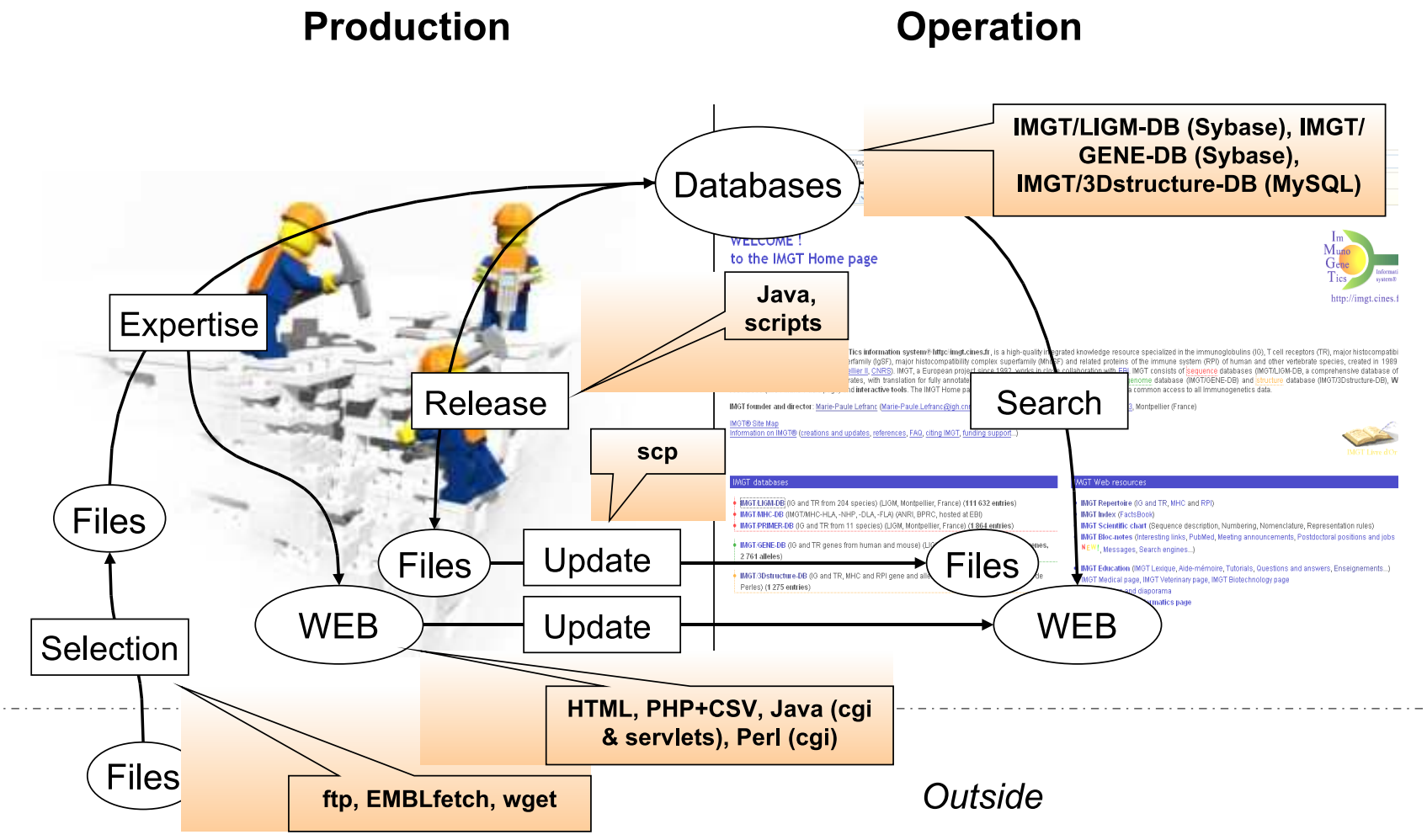


Objectifs

- Maintenir une cohérence scientifique des données exploitées ou crées
- Garantir une expertise biologique adéquate
- Faciliter les échanges
- Gérer les dépendances
- Organiser les révisions
- Obtenir une traçabilité
- Introduire des indicateurs et statistiques
- S'orienter vers une «démarche qualité»

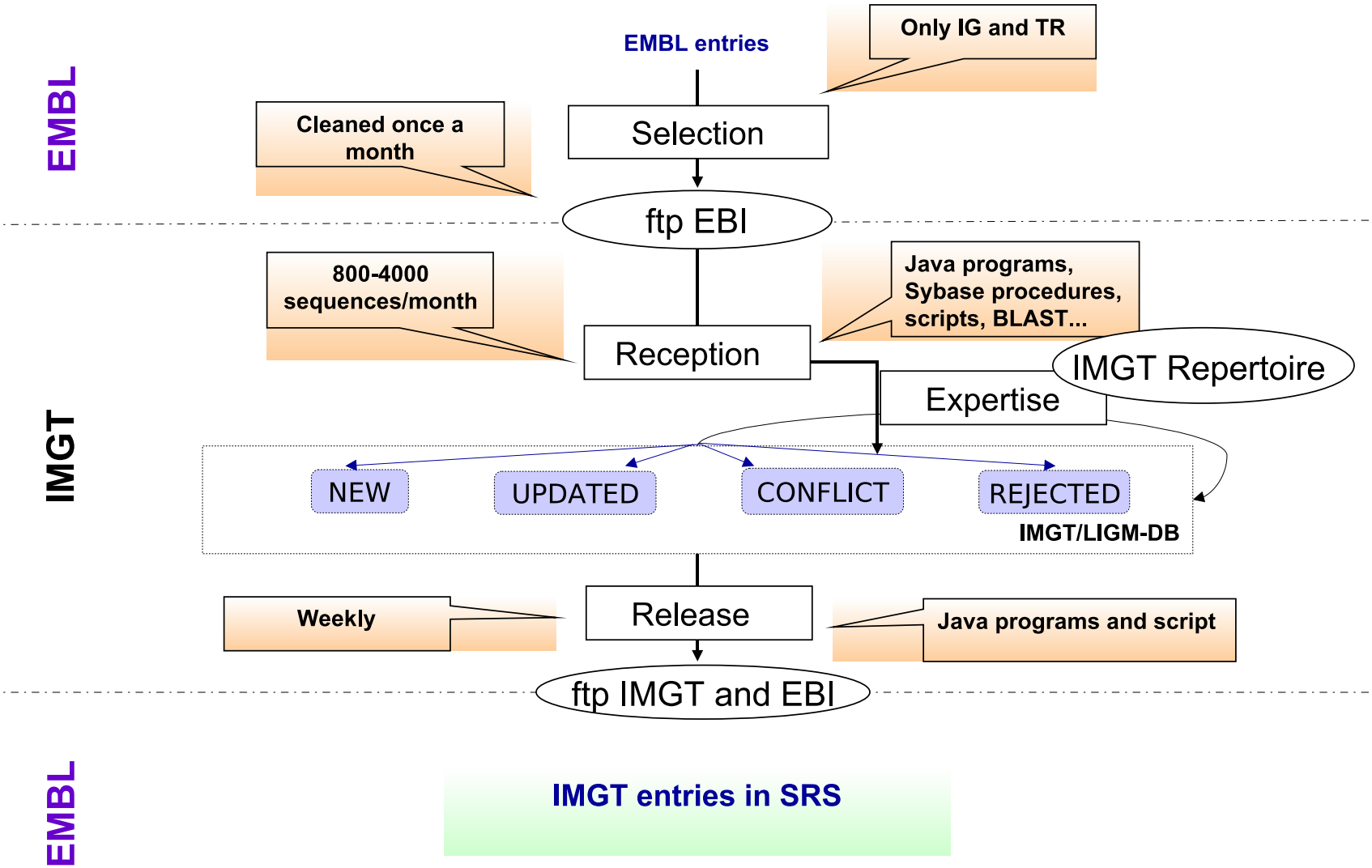
IMGT data-flow

Data-processing infrastructure within IMGT®

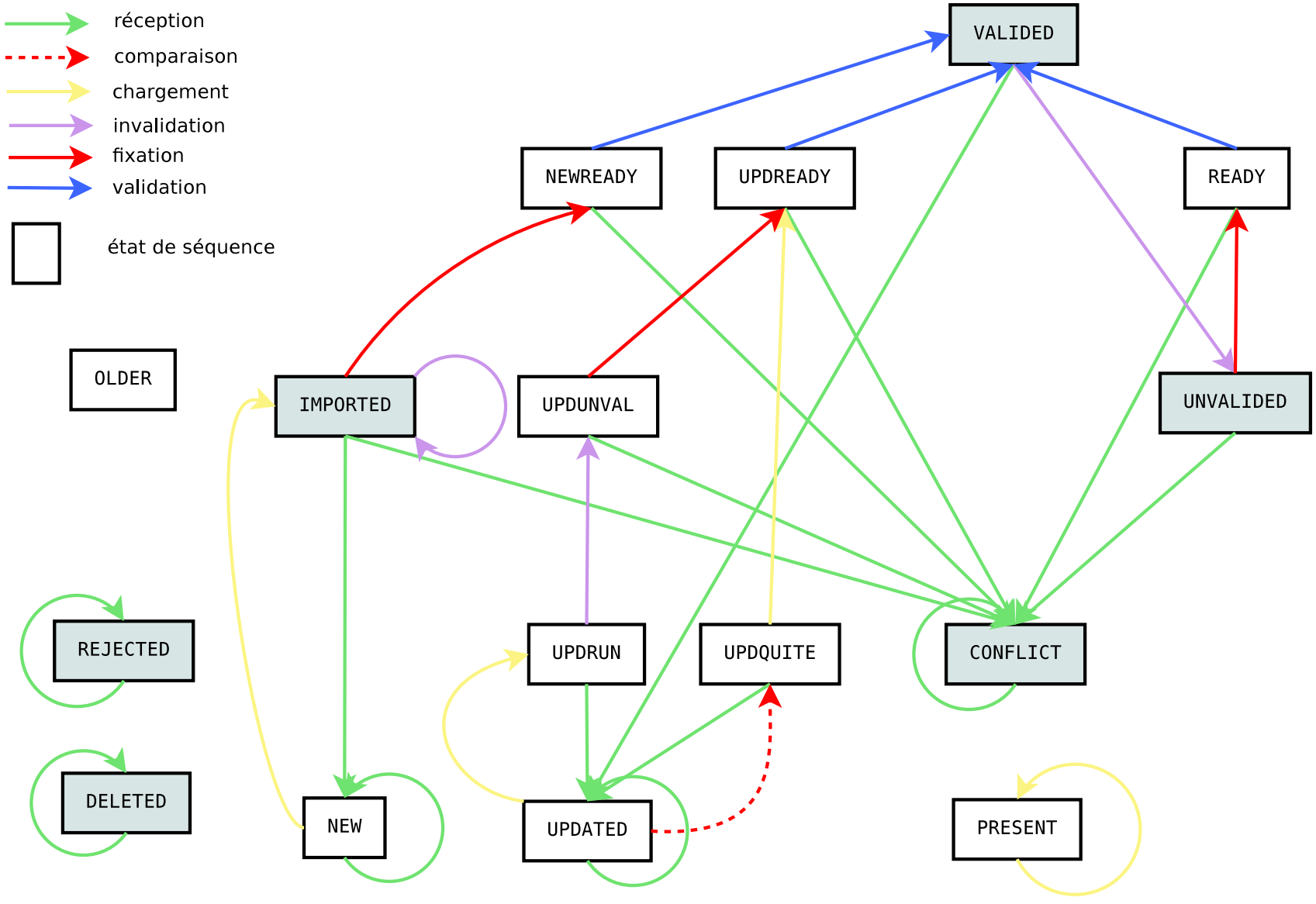


EMBL EBI et IMGT/LIGM-DB

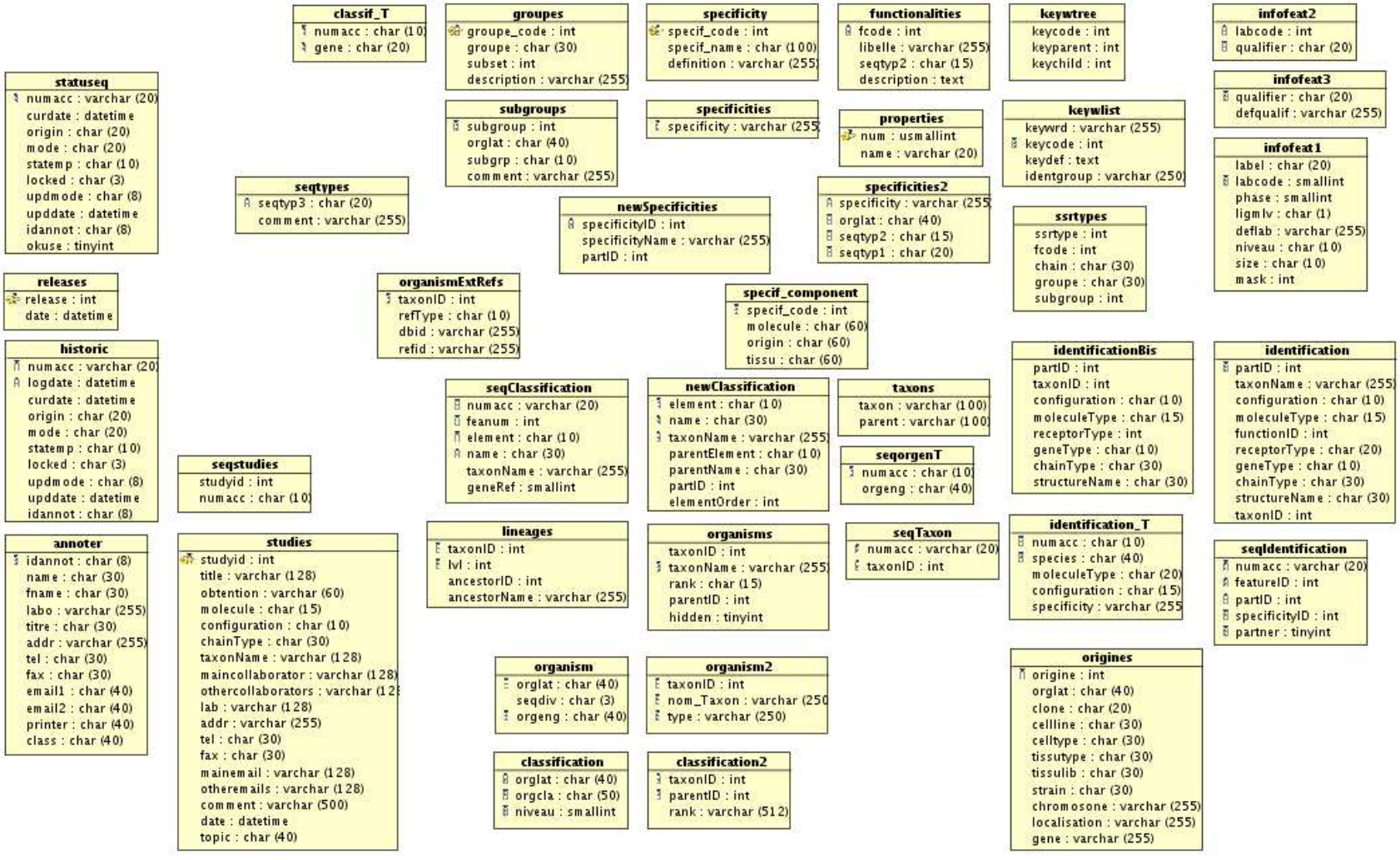
IMGT/LIGM-DB (with EMBL)



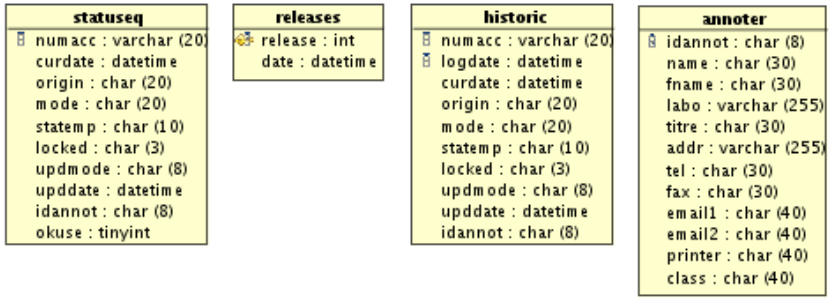
Etat des séquences



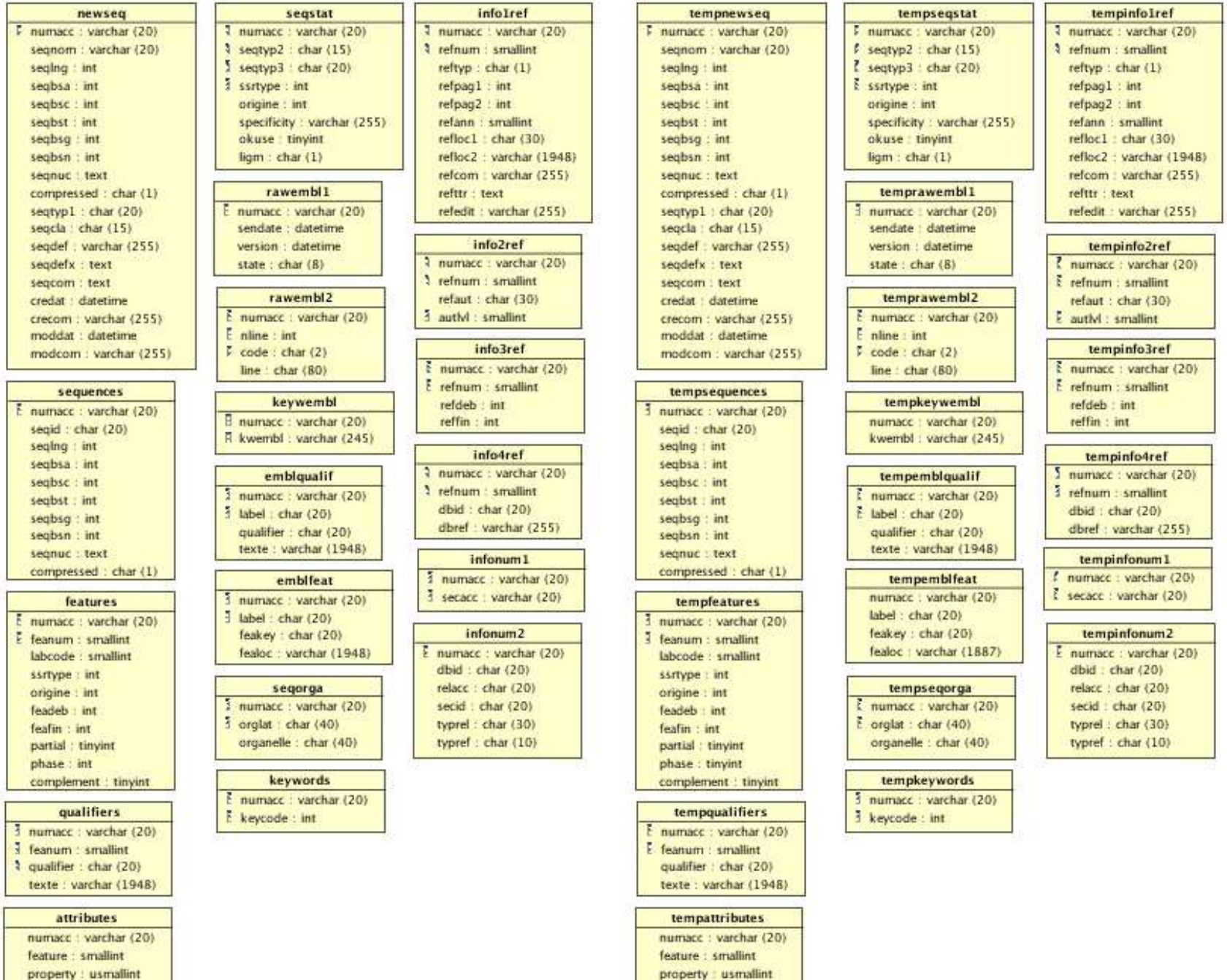
Modèle des connaissances



Modèle des méta-données



Modèle des séquences



Algèbre relationnelle

Introduction

1960 Systèmes de gestion de fichiers/fiches

1970 Systèmes hiérarchiques et réseaux

1980 Systèmes relationnels commerciaux

1990 Systèmes à objets

3 grands leaders des systèmes relationnels commerciaux : Oracle, Informix et Sybase. Mais aussi IBM avec DB2 et Microsoft avec Access.

Introduction

Edgar F. CODD (1970) : définir une algèbre pour formaliser mathématiquement les traitements applicables à des bases de données $\Longrightarrow$ modèle de base de données relationnelle.

- Modèle simple avec peu de concepts et facile à appréhender pour "percevoir" et "implanter" les données.
- Représentation de l'information dans un ensemble de relations (la base).
- Langage commun de définition et de manipulation des données : le SQL.
- Théorie mathématique sous-jacente à ce langage.

Références bibliographiques

- Georges Gardarin, *Bases de données*, Eyrolles, 1999.
- Jean-Luc Hainaut, *Bases de données et modèles de calcul*, Dunod, 2000.
- Jeffrey D. Ullman, *Principles of Database and Knowledge-Base Systems. Volume I*, Computer Science Press, 1988.

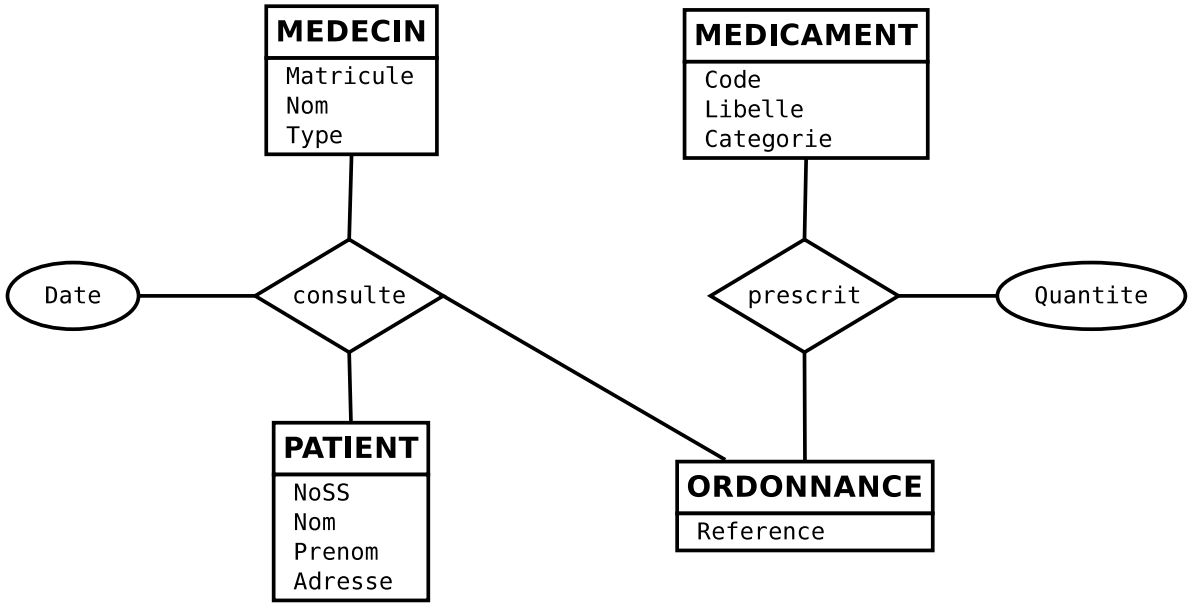
Exemple

Traiter l'information relative à un centre médical. Notamment, pouvoir retrouver des données en répondant à des questions (dites *requêtes*) comme :

- Quel est le numéro de sécurité sociale de M. Amil ?
- Quel(s) médecin(s) a-t-il consulté ?
- Quelle est l'ordonnance de sa dernière consultation ?
- Qui avait-il de prescrit ?
- ...

Exemple

De l'approche entités et associations :



Remarque : diagramme brut sans cardinalité, ni clef.

Exemple

Aux données tabulées :

MEDECIN

Matricule	Nom	Type
271	Dubois	Généraliste
865	Malchausse	Dentiste

PATIENT

NoSS	Nom	Prenom	Adresse
1720833245080	Amil	Jérôme	Nîmes
2740284123456	Brand	Emilie	Nîmes

Exemple

Avec le schéma relationnel :

$$\text{Entités} \rightarrow \begin{cases} \text{MEDECIN(Matricule,Nom,Type)} \\ \text{PATIENT(NoSS,Nom,Prenom,Adresse)} \\ \text{ACTE(Reference,Date)} \\ \text{MEDICAMENT(Code,Libelle,Categorie)} \end{cases}$$

$$\text{Associations} \rightarrow \begin{cases} \text{CONSULTE(Reference,Matricule,NoSS)} \\ \text{PRESCRIT(Reference,Code,Quantite)} \end{cases}$$

Remarque : ORDONNANCE = ACTE $\bowtie$ CONSULTE, soit ORDONNANCE(No,Matricule,NoSS,Date).

Modèle Relationnel

- Fondée sur la théorie des ensembles.
- Relation : représente une collection (ensemble ou multi-ensemble) de données homomorphes (comme dans une table, voire un fichier = collection de fiches).
- Attribut : semblable à une colonne de la table, décrit un domaine.
- N-uplet : semblable à une ligne de la table, décrit une donnée.
- Algèbre : structure d'ensemble de relations muni de lois de composition interne $\Longrightarrow$ langage et égalité.

Langage Relationnel

Attributs
Exemple : Nom, Prenom, DateNaissance, ...
Un symbole parmi un ensemble $\mathcal{A} = \{A_1, \ldots, A_m, \ldots\}$
Domaines
Exemple : texte, nombre, date, ...
Un symbole parmi un ensemble $\mathcal{D} = \{D_1, \ldots, D_n, \ldots\}$ souvent fini et prédéfini (suivant les types de données retenus pour le système)
Chaque domaine D_i est un ensemble (fini ou non) de valeurs, ex. "Jérôme", 12, 10-01-2002, etc.
Remarque : les D_i sont non nécessairement disjoints.

Langage Relationnel

Relations

- Deux formes : intention (variable) et extension (n-uplet)
- Dénotent des collections de données
- Chaque relation est définie sur un *schéma* qui est l'ensemble des attributs qui les composent
- La *dimension* d'une relation est le cardinal de son schéma
- Base de données relationnelles : ensemble de relations

Relations en intention

Des variables

- Syntaxe : R
- Déclaration : $R(A_1 : D_1, \ldots, A_n : D_n)$
- Condition : les A_i tous distincts deux à deux ($A_i \in \mathcal{A}$)
- Sémantique : dénote un sous-ensemble de $D_1 \times \cdots \times D_n$ ($D_i \in \mathcal{D}$)
- Schéma : $R^+ = \{A_1, \ldots, A_n\}$
- Composante : $dom_R(A_i) = D_i$

Exemple : MEDECIN, PATIENT, ORDONNANCE, etc.

Relations en extension

à valeur un ensemble de N-uplets

- Arité : nulle
- Syntaxe : $(A_1 : v_1, \ldots, A_n : v_n)$
- Condition : pour les $A_i \in \mathcal{A}$, les $v_i \in D_j$
- Sémantique : dénote une relation atomique ayant pour n-uplets $\{t\}$
- Schéma : $t^+ = \{A_1, \ldots, A_n\}$

Exemple : (NoSS : 1720833245080, Nom : Amil, Prenom : Jérôme, Adresse : Nîmes)

Propriétés

R-compatibilité : un n-uplet $(A_1 : v_1, \ldots, A_n : v_n)$ et une relation R ont le même schéma et les $v_i \in dom_R(A_i)$

Exemple : (NoSS : 1720833245080, Nom : Amil, Prenom : Jérôme, Adresse : Nîmes) est PATIENT-compatible

Schéma-compatibilité : deux relations qui ont le même schéma et les même domaines associés

Remarque : l'ordre des composants d'un n-uplet n'est pas important avec la notion d'attribut

Relations à attributs $\Longrightarrow$ relations classiques

Langage Relationnel

Opérations

Des classiques sur les ensembles à d'autres plus spécifiques :

- l'union $\cup$,
- l'intersection $\cap$,
- la différence $\setminus$,
- le produit cartésien $\times$.
- le renommage ρ,
- la projection π,
- la sélection σ,
- les jointures $\bowtie$.

Union

Formellement

- Arité : binaire
- Syntaxe : $R_1 \cup R_2$
- Condition : R_1 et R_2 schéma-compatible
- Sémantique : $R_1 \cup R_2 = \{t | t \in R_1 \vee t \in R_2\}$
- Schéma : $(R_1 \cup R_2)^+ = R_1^+ = R_2^+$

Intersection

Formellement

- Arité : binaire
- Syntaxe : $R_1 \cap R_2$
- Condition : R_1 et R_2 schéma-compatible
- Sémantique : $R_1 \cap R_2 = \{t | t \in R_1 \wedge t \in R_2\}$
- Schéma : $(R_1 \cap R_2)^+ = R_1^+ = R_2^+$

Différence

Formellement

- Arité : binaire
- Syntaxe : $R_1 \setminus R_2$
- Condition : R_1 et R_2 schéma-compatible
- Sémantique : $R_1 \setminus R_2 = \{t | t \in R_1 \wedge t \notin R_2\}$
- Schéma : $(R_1 \setminus R_2)^+ = R_1^+ = R_2^+$

Renommage (I)

Formellement

- Arité : unaire
- Syntaxe : $\rho_X(R)$
- Condition : $X = \{\ldots A_i/B_i \ldots\}$ où $A_i \in R^+$ et $B_i \in \mathcal{A}$
- Sémantique : $\rho_{A_1/B_1,\ldots,A_n/B_n}(R) = \{(\rho_(A_1) : v_1, \ldots, \rho_(A_n) : v_n) | (A_1 : v_1, \ldots A_n : v_n) \in R\}$ où $\rho_X(A_i) = B_i$ si $(A_i/B_i) \in X$, $\rho_X(A_i) = A_i$ sinon
- Schéma : $(\rho_X(R))^+ = \{\rho_X(A_1), \ldots, \rho_X(A_n)\}$
- Remarque : existe aussi avec un renommage de la relation $\rho_{R_2}(R_1)$

Renommage (II)

Exemple

$\rho_{Nom/NomPatient}(\text{PATIENT})$

NoSS	NomPatient	Prenom	Adresse
1720833245080	Amil	Jérôme	Nîmes
2740284123456	Brand	Emilie	Nîmes

Projection (I)

Intuitivement
Extraire des colonnes
Formellement

- Arité : unaire
- Syntaxe : $\pi_X(R)$ ou $R[X]$
- Condition : $X \subseteq R^+$
- Sémantique : $\pi_X(R) = \{(\ldots, A_i : v_i, \ldots) | (A_1 : v_1, \ldots, A_n : v_n) \in R \wedge A_i \in X\}$
- Schéma : $(\pi_X(R))^+ = X$
- Remarque : $|(\pi_X(R))^+| \leq |R^+|$

Projection (II)

Exemple

$\pi_{NoSS,Nom}(\text{PATIENT})$

NoSS	Nom
1720833245080	Amil
2740284123456	Brand

Sélection (I)

Intuitivement
Extraire des lignes
Formellement

- Arité : unaire
- Syntaxe : $\sigma_F(R)$ ou $R\{F\}$
- Condition : F est une formule logique sur le R^+ telle que pour tout $t \in R$, $F(t)$ est vrai ou faux
- Sémantique : $\sigma_F(R) = \{t | t \in R \wedge F(x)\}$
- Schéma : $(\sigma_F(R))^+ = R^+$
- Remarque : $\sigma_F(R) \subseteq R$

Sélection (I)

Exemple

$$\sigma_{Nom=Amil}(\text{PATIENT})$$

NoSS	Nom	Prenom	Adresse
1720833245080	Amil	Jérôme	Nîmes

Jointures (I)

Produit (cartésien)

- Arité : binaire
- Syntaxe : $R_1 \times R_2$
- Condition : $R_1^+ \cap R_2^+ = \emptyset$ sinon renommage préalable des attributs communs de R_1 et R_2 (préfixés par la relation)
- Sémantique :
  $R_1 \times R_2 = \{(A_1 : v_1, \ldots, A_m : v_m, B_1 : w_1, \ldots, B_n : w_n) | (A_1 : v_1, \ldots, A_m : v_m) \in R_1 \wedge (B_1 : w_1, \ldots, B_n : w_n) \in R_2\}$
- Schéma : $(R_1 \times R_2)^+ = R_1^+ \cup R_2^+$

Jointures (II)

Exemple

PATIENT2 $\times$ MEDECIN

NoSS	NomP	Prenom	Adresse	Matricule	Nom	Type
1720833245080	Amil	Jérôme	Nîmes	271	Dubois	Généraliste
1720833245080	Amil	Jérôme	Nîmes	865	Malchausse	Dentiste
2740284123456	Brand	Emilie	Nîmes	271	Dubois	Généraliste
2740284123456	Brand	Emilie	Nîmes	865	Malchausse	Dentiste

Jointures (III)

Theta-Jointure

- Arité : binaire
- Syntaxe : $R_1 \bowtie_\Theta R_2$
- Condition : Θ est une condition
- Sémantique : $R_1 \bowtie_\Theta R_2 = \{t | t \in R_1 \times R_2 \wedge \Theta(t)\}$
- Schéma : $(R_1 \bowtie_\Theta R_2)^+ = R_1^+ \cup R_2^+$
- Remarque : si $R_1^+ \cap R_2^+ =$ alors on a $R_1 \bowtie_\Theta R_2 = R_1 \times R_2$

Remarque : équi-jointure si Θ de la forme $R_1.A = R_2.B$

Jointures (IV)

Jointure naturelle

- Arité : binaire
- Syntaxe : $R_1 \bowtie R_2$
- Sémantique : dénote une équi-jointure avec une égalité entre paires d'attributs communs de R_1 et R_2 (équi-jointure dite naturelle)
- Remarque : cf. Theta-jointure

Égalités

Expriment des propriétés des opérateurs algébriques comme par exemple la commutativité du produit ainsi :

- $R \times S = S \times R$
- $R \times (S \times T) = (R \times S) \times T$
- $\pi_Y(\pi_X(R)) = \pi_Y(R)$ si $Y \subseteq X$
- $\pi_Z(R \times S) = \pi_X(R) \times \pi_Y(S)$ où $X = Z \cap R^+$ et $Y = Z \cap S^+$
- $\pi_X(R \cup S) = \pi_X(R) \cup \pi_X(S)$

Égalités

- $\sigma_{F_1}(\sigma_{F_2}(R)) = \sigma_{F_1 \wedge F_2}(R)$
- $\sigma_F(R \cup S) = \sigma_F(R) \cup \sigma_F(S)$
- $\sigma_F(R \setminus S) = \sigma_F(R) \setminus \sigma_F(S)$
- $\sigma_F(R_1 \times R_2) = \sigma_{F_1}(R_1) \times \sigma_{F_2}(R_2)$ où $F = F_1 \wedge F_2$ et F_i porte sur R_i (inclus le cas vide pour un F_i)
- $\pi_X(\sigma_F(R)) = \sigma_F(\pi_X(R))$ si F porte sur X sinon $\pi_X(\sigma_F(R)) = \pi_X(\sigma_C(\pi_{YX}(R)))$ où $Y = var(F) \setminus X$

Pour résumer

- Chaque opération détermine une relation (clos)
- Elles peuvent être composées (liberté)

$\Longrightarrow$ détermine une Algèbre dite Relationnelle

- Terme (expression du langage algébrique) :

$$T \doteq R|t|(T \cup T)|(T \cap T)|(T \setminus T)$$

$$|\rho(T)|\pi_X(T)|\sigma_F(T)|(T \times T)|(T \bowtie T)$$

- Requêtes = termes de l'algèbre relationnelle

Vers SQL

Intuitivement :

$$\pi_X(\sigma_F(R)) \Longleftrightarrow \begin{cases} \texttt{select}\ X \\ \texttt{from}\ R \\ \texttt{where}\ F \end{cases}$$

à des transformations des termes X, R et F près.
Exemple : si en SQL, $R = R_1, \ldots, R_l$, alors en Algèbre Relationnelle $R = \rho_?(R_1) \times \cdots \times \rho_?(R_l)$

Vers SQL

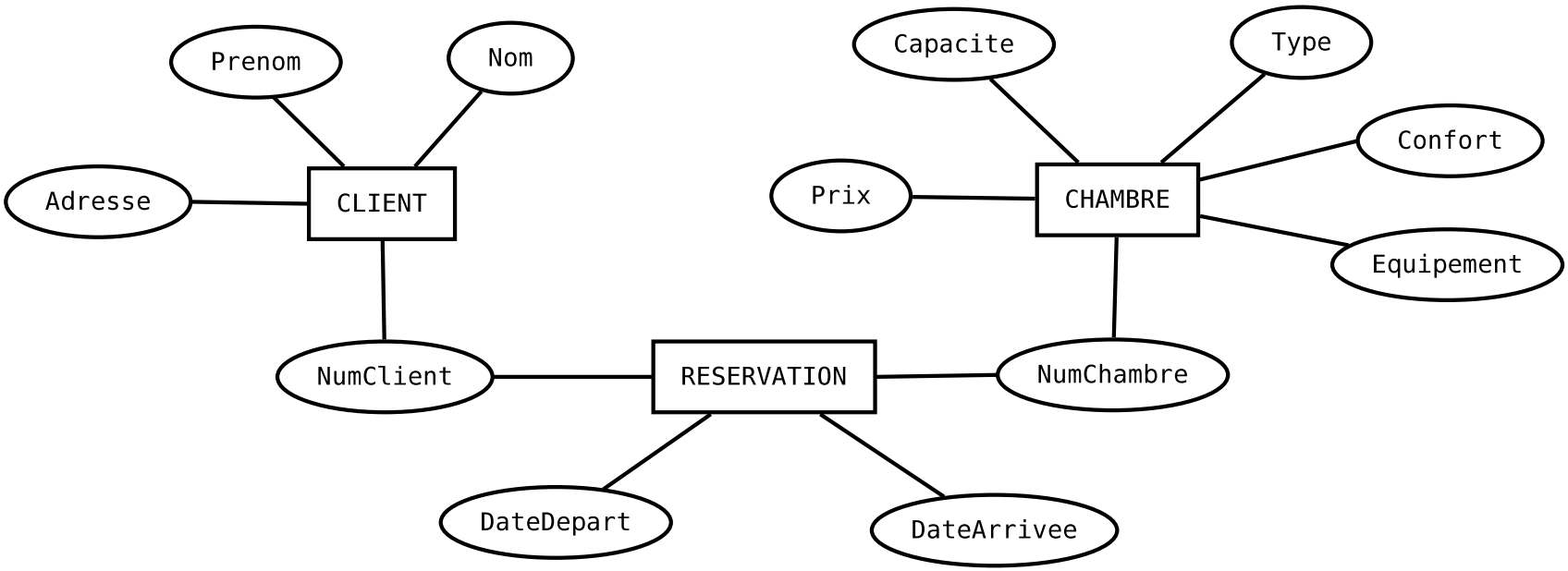
- Extension sémantique : ensembles $\rightarrow$ multi-ensembles
- Opérations étendues : classiques + projection, sélection, jointure
- Remarques : certaines propriétés sont perdues, ex. $R \cap (S \cup T) \neq (R \cap S) \cup (R \cap T)$
- Opérations supplémentaires :
  - élimination des duplicatas δ,
  - trie $\tau_X(R)$ (selon les attributs de X),
  - agrégation $\phi(X)$ avec ϕ somme, moyenne, cardinal, min, max, etc.
  - groupement $\gamma_X(R)$ (avec agrégations).

Conclusion

Conséquences

- Plusieurs algèbres équivalentes suivants le choix des primitives (opérations de base)
  - $R \cap S = R \setminus (R \setminus S)$
  - $R \bowtie_\Theta S = \sigma_\Theta(R \times S)$
  - $R \bowtie S = \pi_?(R \bowtie_? S)$
- Algèbre "standard" : SPC (Sélection/Projection/produit Cartésien)
- Facilité la formulation et/ou l'évaluation des requêtes $\Longrightarrow$ optimisations
- Langage trop libre $\Longrightarrow$ contraintes

Exercice : chambres d'hôtel



Exercice : questions

1. Les chambres avec bain et télévision ?
2. Les numéros des chambres et leur capacité ?
3. Les noms de clients ayant réservés une chambre pour le 25/12/2001 ?
4. Le nom des clients et le confort des chambres qu'ils ont réservés ?
5. La capacité théorique d'accueil de l'hôtel ?

Exprimer ces requêtes en algèbre relationnelle.

Modèle Relationnel

- Objectif : éviter les inaptitudes/répétitions/pertes d'information.
- Incovénient : système AR trop libre $\Longrightarrow$ ajouter des formes de contraintes.
- Problème : comparer plusieurs schémas relationnels équivalents pour une même base (ensemble de données) car certains sont plus «efficaces» que d'autres pour l'ajout, la modification et la suppression.
- Une solution : ajouter des dépendances fonctionnelles au schéma

Exemple

PATIENT

NoSS	Nom	Prenom	Adresse
1720833245080	Amil	Jérôme	Nîmes
2740284123456	Brand	Emilie	Nîmes

Est-il possible d'ajouter le n-uplet (NoSS : 1720833245080, Nom : Augier, Prenom : Rémi, Adresse : Montpellier) ? Oui ?

Non ? Notion d'attributs identifiants ou clefs (ici NoSS)

Dépendances Fonctionnelles

Informellement
Dépendance fonctionnelle : attributs qui déterminent d'autres attributs

Sur-clef : cas particulier où tous les attributs sont déterminés

Clef : plus petit ensemble pour l'inclusion (minimalité)

Notation : $\underline{A}$ désigne un attribut de la clef

Exemple

Schéma relationnel

$$\text{Entités} \rightarrow \begin{cases} \text{MEDECIN}(\underline{\text{Matricule}},\text{Nom},\text{Type}) \\ \text{PATIENT}(\underline{\text{NoSS}},\text{Nom},\text{Prenom},\text{Adresse}) \\ \text{ACTE}(\underline{\text{Reference}},\text{Date}) \\ \text{MEDICAMENT}(\underline{\text{Code}},\text{Libelle},\text{Categorie}) \end{cases}$$

$$\text{Associations} \rightarrow \begin{cases} \text{CONSULTE}(\underline{\text{Reference}},\text{Matricule},\text{NoSS}) \\ \text{PRESCRIT}(\underline{\text{Reference}},\underline{\text{Code}},\text{Quantite}) \end{cases}$$

Dépendances Fonctionnelles

Cadre Unique relation R sur les attributs $\mathcal{A}$ (Schéma *universel* d'une base)

Langage

- Syntaxe : $X \rightarrow Y$ où X et Y sont des parties de $\mathcal{A}$
- Sémantique : pour toute instance de R et pour tous n-uplets t_1 et t_2 tels que $\pi_X(t_1) = \pi_X(t_2)$, on a : $\pi_Y(t_1) = \pi_Y(t_2)$

Dépendances Fonctionnelles

Modèle

- si $XY = \mathcal{A}$ alors deux n-uplets t et t' coïncidant sur X sont identiques
- plusieurs DF sur un schéma $(R, \mathcal{A}) \Longrightarrow$ schéma $(R, \mathcal{A}, \mathcal{F})$ où $\mathcal{F}$ est un ensemble de DF

Question Pour une même relation, plusieurs ensembles de DF possible avec équivalence

Dépendances Fonctionnelles

Définitions

- *Conséquence logique* : si $X \rightarrow Y$ est valide pour toute instance de $(R, \mathcal{A}, \mathcal{F})$
- *Fermeture transitive* : ensemble de DF noté $\mathcal{F}^+$ qui contient toutes les conséquences logiques de $\mathcal{F}$
- *Équivalence* : deux ensembles de DF $\mathcal{F}$ et $\mathcal{F}'$ sont *équivalents* si et seulement si $\mathcal{F}^+ = \mathcal{F}'^+$

Théorème Soit $\mathcal{F}$ un ensemble de DF, la dépendance $X \rightarrow Y$ est dérivée de $\mathcal{F}$ ($X \rightarrow Y \in \mathcal{F}^+$) si et seulement si $Y \in X^+/\mathcal{F}$.

Fermeture Transitive

Algorithme Fermeture transitive de X par rapport à $\mathcal{F}$

- **Entrées** : X ensemble d'attibuts de $(R, \mathcal{A}, \mathcal{F})$
- **Sortie** : $X^+/\mathcal{F}$
- **Instructions** :

$X^+ := X$;
$\mathcal{F}' := \mathcal{F}$;
tant que $\exists Y \to Z \in \mathcal{F}' | Y \subseteq X^+$
faire
$\mathcal{F}' := \mathcal{F}' \setminus \{Y \to Z\}$;
$X^+ := X^+ \cup Z$;

Conséquence Logique

Solution W. W. Armstrong (1974) : système (logique) déductif pour "établir" les conséquences logiques d'un ensemble de DF

	Nom	Conclusion	Hypothèse
Règles	Réflexivité	$X \to Y$	si $Y \subseteq X$
	Augmentation	$XZ \to YZ$	si $X \to Y$ et $Z \in \mathcal{A}$
	Transitivité	$X \to Z$	si $X \to Y$ et $Y \to Z$

Système d'Armstrong

Définitions Un système déductif est dit :

- *correct* s'il n'engendre que des formules (ici, DF) valables.
- *complet* s'il permet d'obtenir toute formule (ici, DF) valables.
- *adéquat* si et seulement si il est correct et complet.

Théorème Le système d'Armstrong est adéquat.

Couverture Irredondante Minimale

Objectif : Bon ensemble de DF (éliminer la redondance)
Définitions

- $A \in X$ est un *attribut redondant* dans $X \rightarrow Y$ de $\mathcal{F}$ si et seulement si $Y \in (X \setminus \{A\})^+)$ par rapport à $\mathcal{F}$, (DF minimale à gauche)
- $X \rightarrow Y$ est une *DF redondante* dans $\mathcal{F}$ si et seulement si $(\mathcal{F} \setminus \{X \rightarrow Y\})^+ = \mathcal{F}^+$ ou $Y \in X^+$ par rapport à $\mathcal{F} \setminus \{X \rightarrow Y\}$,
- $\mathcal{F}'$ est une CIM de $\mathcal{F}$ si et seulement si $\mathcal{F}$ et $\mathcal{F}'$ équivalents et les DF de $\mathcal{F}'$ telles que :
  1. sans d'attribut redondant (en partie droite),
  2. pas redondante.

$\Longrightarrow$ algorithme de calcul d'une CIM

Clef (retour)

Définitions

- Sur-clef K d'un schéma avec DF (R, U, F) ssi $K \to U$ est une DF dérivable
- Clef K est une sur-clef et minimale (ne contient pas strictement une autre sur-clef) ie. $K \to \mathcal{A}$ est minimale à gauche

Problème
Calculer les clefs d'un schéma $(R, \mathcal{A}, \mathcal{F})$
Remarque : $\mathcal{A}$ est une sur-clef triviale
$\Longrightarrow$ Un algorithme : minimisation de la DF $\mathcal{A} \to \mathcal{A}$

Décomposition

Objectif
Casser un schéma $(R, \mathcal{A})$ en plusieurs schémas $(R_i, \mathcal{A}_i)$ sans perte d'information
Définition
Décomposition : les $(R_i, \mathcal{A}_i)$ sont des sous-schémas
Propriétés

- $\mathcal{A} = \bigcup_i \mathcal{A}_i$ (adéquation des attributs)
- $R = \pi_{\mathcal{A}_1}(R) \bowtie \cdots \bowtie \pi_{\mathcal{A}_n}(R)$ (adéquation des N-uplets)

Bonne décomposition

Problème
Décomposition respectant les DF ?
Exemple
Schéma $(R, ABCD, \{AB \to C, C \to D, D \to A\})$ décomposé en deux sous-schémas $(R_1, ABC, \mathcal{F}_1)$ et $(R_2, AD, \mathcal{F}_2)$. Que représentent $\mathcal{F}_1$ et $\mathcal{F}_2$?
A priori, seule $AB \to C$ s'applique sur R_1, mais en fait $C \to A$ est déductible sur R et s'applique aussi sur R_1.

Remarque : complexité exponentielle dans le pire des cas !

Normalisation

Objectif

Normaliser afin de proposer un schéma relationnel "propre" qui soit référentiel et permette d'éviter :

- la redondance des informations,
- les incohérences dans les mises à jour,
- les anomalies lors des insertions/suppressions.

Plusieurs formes normales selon les critères optimisés : 1FN, 2FN, 3FN, FBCN, 4FNF et 5FN.

Formes Normales

Première FN
Toutes les données sont représentées dans des relations binaires (clef, valeur)

Tout est atomique : attribut simple ou monovalué

Toujours le cas dans ce modèle relationnel (pas de relation de second ordre)

Formes Normales

Deuxième FN
$(R, \mathcal{A}, \mathcal{F})$ est en 2FN si et seulement si tout attribut de $\mathcal{A}$ n'appartenant pas à une clef dépend directement d'une clef.

Exemple : Schéma (R,{Appareil, Vol, AéroportDépart, AéroportArrivée},{Vol $\rightarrow$ Appareil,{Vol,AéroportDépart} $\rightarrow$ AéroportArrivée})
Unique clef : K = {Vol, AéroportDépart}
Vol, AéroportDépart $\in$ K. AéroportArrivée dépend directement de K. Mais Appareil ne dépend que de Vol et non de K directement. Ce schéma n'est pas en 2FN.

Formes Normales

Troisième FN
$(R, \mathcal{A}, \mathcal{F})$ avec $\mathcal{F}$ CIM est en 3FN ssi pour toute DF $X \rightarrow A$ de $\mathcal{F}$, on a :

- X est une clef,
- A appartient à une clef.

Forme normale de Boyce-Codd (BCNF)
$(R, \mathcal{A}, \mathcal{F})$ avec $\mathcal{F}$ CIM est en BCNF ssi est en 3FN et pour toute DF $X \rightarrow Y$ de $\mathcal{F}$, X est un clé

Formes Normales

Algorithme (dû à Philip A. Bernstein dans les années 70)

- **Entrée** : un schéma $(R, \mathcal{A}, \mathcal{F})$
- **Sortie** : un ensemble de schémas $\{(R_i, \mathcal{A}_i, \mathcal{F}_i)\}$ en 3FN
- **Instructions** :
  1. $\mathcal{F}' = \mathit{CIM}(F)$;
  2. $\mathcal{K} = \mathit{Clef}(F')$;
  3. Partitionner $\mathcal{F}'$ selon les DF ayant même partie gauche et construire les schémas maximaux pour l'inclusion

$$(R_i, \mathcal{A}_i = \mathit{Max}(X \cup Y), \{X_j \rightarrow Y_j | X_j \cup Y_j \subseteq \mathcal{A}_i\}$$

Formes Normales

- **Instructions** :
  4. Si aucune clef ne figure dans les schémas précédents, ajouter le schéma

$$(R_{i+1}, K, \{K \rightarrow K\})$$

Conclusion

Parfois les DF ne sont pas suffisantes

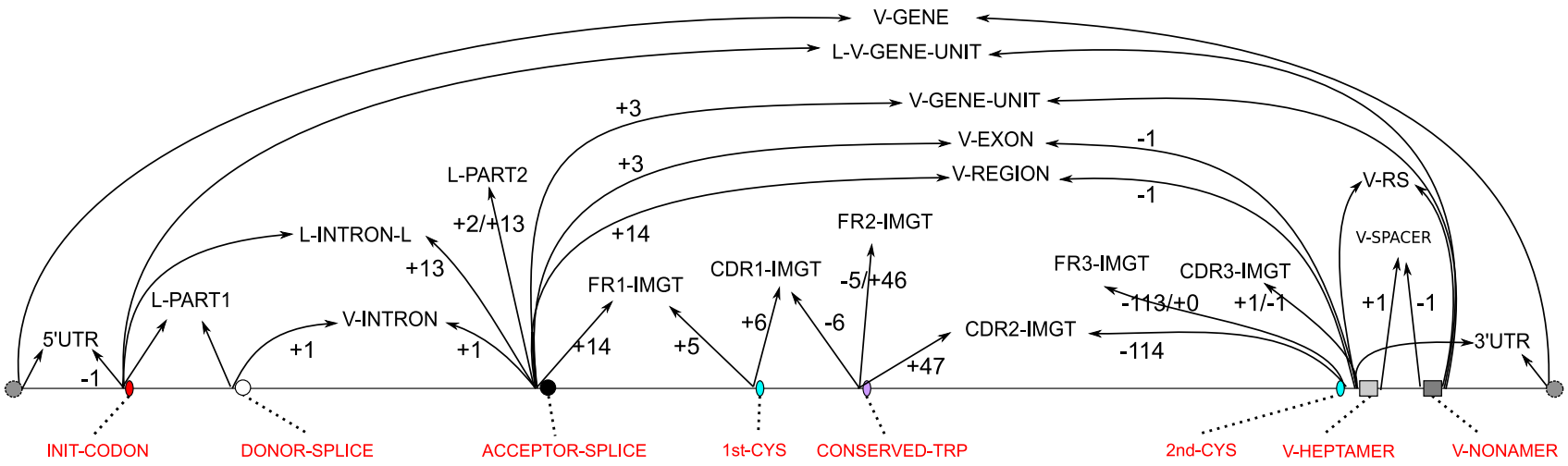
Autres contraintes :

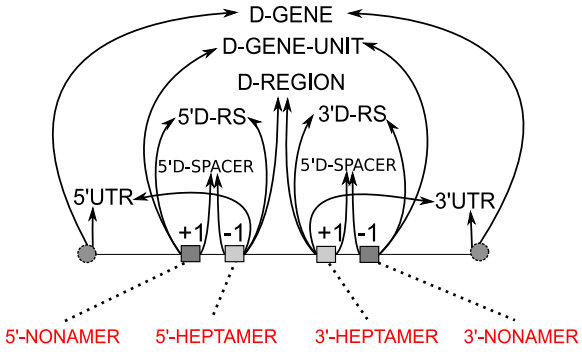
- dépendances multi-valuées (4FN, 5FN)
- d'inclusion
- de jointure
- de cardinalité (unicité)
- etc.

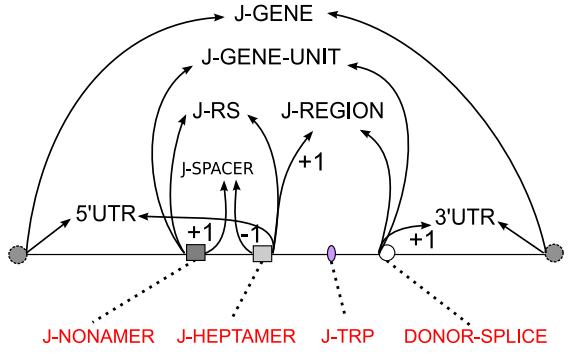
Algèbre d’intervalles

Objectif

Exemple des V,D et J-GENE :







Introduction

Situation ensembliste, deux ensembles E et F :

- $E = F$,
- $E \subset F$,
- $E \cap F = \emptyset$,
- d'autres ?...

Définitions

Une relation d'*ordre* R dans un ensemble E est une relation binaire telle que :

- réflexive : xRx,
- transitive : si xRy et yRz alors xRz,
- antisymétrique : si xRy et yRx alors $x = y$.

Ordre *total* : xRy ou yRx, tous les éléments sont comparables.
Sur les sommets d'un graphe orienté, la relation « être accessible depuis » est un pré-ordre (c'est en fait la fermeture réflexive et transitive du graphe). Si le graphe est sans cycle, cette relation devient un ordre.

Relation d'ordre

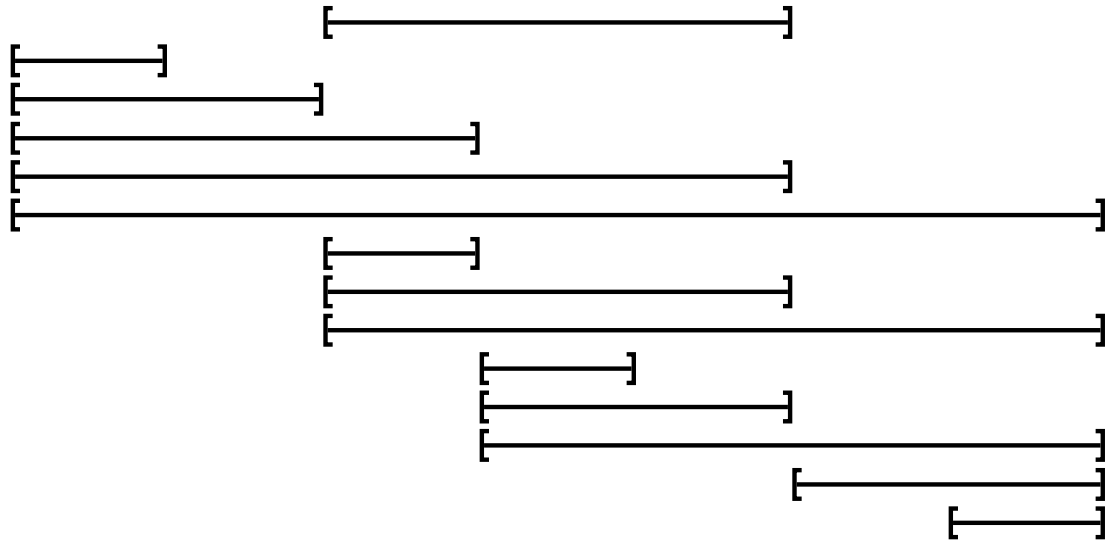
Ordre *strict* : R transitive et irréflexive.
Ordre strict total : ou bien $x < y$, ou bien $y < x$, sinon $x = y$!
Rmq. pas de confusion possible avec le sens total précédent, car une relation d'ordre strict, qui est irréflexive, ne peut être totale au sens où l'est un ordre large.

Intervalle

Étant donné un ensemble E muni d'une relation d'ordre total $<$, un *intervalle* (ou segment) $(a; b)$ est un sous-ensemble de E déterminé par deux éléments a et b tels que $a < b$, et où a (resp. b) dit la *borne inférieure* (resp. *supérieure*).
$(a; b) = \{x | a < x, x < b\}$ Pour un intervalle quelconque A, on désigne par A^- sa borne inférieure et par A^+ sa borne supérieure.
On dit qu'un intervalle A est un *point* si $A^- = A^+$.
Un tel objet est isomorphe à un élément du produit cartésien $E \times E$.

Relations d'intervalles

p	précède
m	rencontre
o	pénètre
fi	
di	couvre
s	débute
$=$	coïncide
si	dépasse
d	
f	termine
oi	sort
mi	relaye
pi	succède



$$\mathcal{RI} \leftarrow \{p, m, o, fi, di, s, =, si, d, f, oi, mi, pi\}$$

Quelques propriétés

Relations inverses :

- $A\ p\ B \iff B\ pi\ A$
- $A\ m\ B \iff B\ mi\ A$
- $A\ o\ B \iff B\ oi\ A$
- $A\ fi\ B \iff B\ f\ A$
- $A\ di\ B \iff B\ d\ A$
- $A\ s\ B \iff B\ si\ A$

Soit donc au plus 7 relations.
On retrouve : $A \subset B \Leftrightarrow A\ d\ B \vee A\ s\ B \vee A\ f\ B$.

Relations aux bornes

- Par définition :

A	p	m	o	fi	di	s	$=$	si	d	f	oi	mi	pi	B
A^-	$<$	$<$	$<$	$<$	$<$	$=$	$=$	$=$	$>$	$>$	$>$	$\star$	$>$	B^-
A^-	$<$	$<$	$<$	$<$	$<$	$<$	$\star$	$\star$	$<$	$\star$	$<$	$=$	$>$	B^+
A^+	$<$	$=$	$>$	$\star$	$>$	$\star$	$\star$	$>$	$>$	$>$	$>$	$>$	$>$	B^-
A^+	$<$	$\star$	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$	$>$	$>$	B^+

- Quel choix pour les cases $\star$?

Cas particulier des «points»

Exemple : $B^- = B^+$

A	p	m	o	fi	di	s	$=$	si	d	f	oi	mi	pi	B
A^-	$<$	$<$		$<$	$<$			$=$				$=$	$>$	$B^- = B^+$
A^+	$<$	$=$		$=$	$>$			$>$				$>$	$>$	$B^- = B^+$

en composant (union) les lignes respectives.

Mais il y a alors des ambiguïtés dues aux équivalences :

- $A\ fi\ B \iff A\ m\ B$,
- $A\ si\ B \iff A\ mi\ B$.

Soit donc 5 relations possibles.

Relations aux bornes (bilan)

A	p	m	o	fi	di	s	$=$	si	d	f	oi	mi	pi	B
A^-	$<$	$<$	$<$	$<$	$<$	$=$	$=$	$=$	$>$	$>$	$>$	$>$	$>$	B^-
A^-	$<$	$<$	$<$	$<$	$<$	$<$	$\leq$	$\leq$	$<$	$\leq$	$<$	$=$	$>$	B^+
A^+	$<$	$=$	$>$	$\geq$	$>$	$\geq$	$\geq$	$>$	$>$	$>$	$>$	$>$	$>$	B^-
A^+	$<$	$<$	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$	$>$	$>$	B^+

Variable

On peut généraliser à un sous-ensemble des relations $\mathcal{RI}$, dit *alternative* et noté R_{AB}, pour décrire les relations possibles entre deux intervalles A et B. On parle de *déterminée* si cet ensemble est réduit à un seul élément, et on peut alors noter indifféremment $A\ \{x\}\ B$ et $A\ x\ B$, autrement (plus d'un élément) on parle d'*indéterminée*. Ainsi, $A\ \{p, m\}\ B$ signifie $A\ \{p\}\ B$ ou $A\ \{m\}\ B$.
Bien sûr, un problème classique consiste à pouvoir alors déterminer toutes les possibilités entre des intervalles étant données des certitudes entre eux.
Par exemple, en considérant $A\ \{d\}\ B$, $B\ \{p, m\}\ C$, et $D\ \{d\}\ C$, on peut montrer que $A\ \{p\}\ D$.
On a : $\subset \equiv \{d, s, f\}$.

Transitivité

Étant donnés trois intervalles A, B et C, le tableau T ci-après donne les 13×13 alternatives de A et C issues des certitudes de A et B, et de B et C.

	p	m	o	fi	di	s
p	p	p	p	p	p	p
m	p	p	p	p	p	m
o	p	p	p, o, m	p, o, m	p, di, o, m, fi	o
fi	p	m	o	fi	di	o
di	p, di, o, m, fi	di, o, fi	di, o, fi	di	di	di, o, fi
s	p	p	p, o, m	p, o, m	p, di, o, m, fi	s
$=$	p	m	o	fi	di	s
si	p, di, o, m, fi	di, o, fi	di, o, fi	di	di	$=, s, si$
d	p	p	p, d, o, m, s	p, d, o, m, s	$\mathcal{RI}$	d
f	p	m	d, o, s	$=, f, fi$	pi, di, oi, mi, si	d
oi	p, di, o, m, fi	di, o, fi	$=, o, oi, s, si, f, fi$	di, oi, si	pi, di, oi, mi, si	d, oi, f
mi	p, di, o, m, fi	$=, s, si$	d, oi, f	mi	pi	d, oi, f
pi	$\mathcal{RI}$	pi, d, oi, mi, f	pi, d, oi, mi, f	pi	pi	pi, d, oi, mi, f

Fermeture

Étant données deux variables R_{AB} et R_{BC}, on peut calculer la variable R_{AC} des relations possibles entre A et C par l'algorithme suivant :

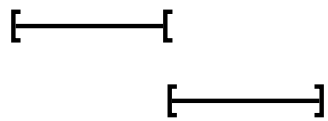
- entrée : R_{AB} et R_{BC}
- sortie : R_{AC}
- instruction :
  $R_{AC} \leftarrow \emptyset$;
  $\forall r \in R_{AB}$:
    $\forall s \in R_{BC}$:
      $R_{AC} \leftarrow R_{AC} \cup T(r, s)$;

Étant donnés un ensemble d'intervalles et des relations possibles entre eux, il est possible de calculer l'ensemble de toutes les alternatives possibles.

Intervalles dans les séquences

Intervalles = segments sur une séquence biologique (chaîne de caractères), soit des éléments de $\mathbb{N}^2$.

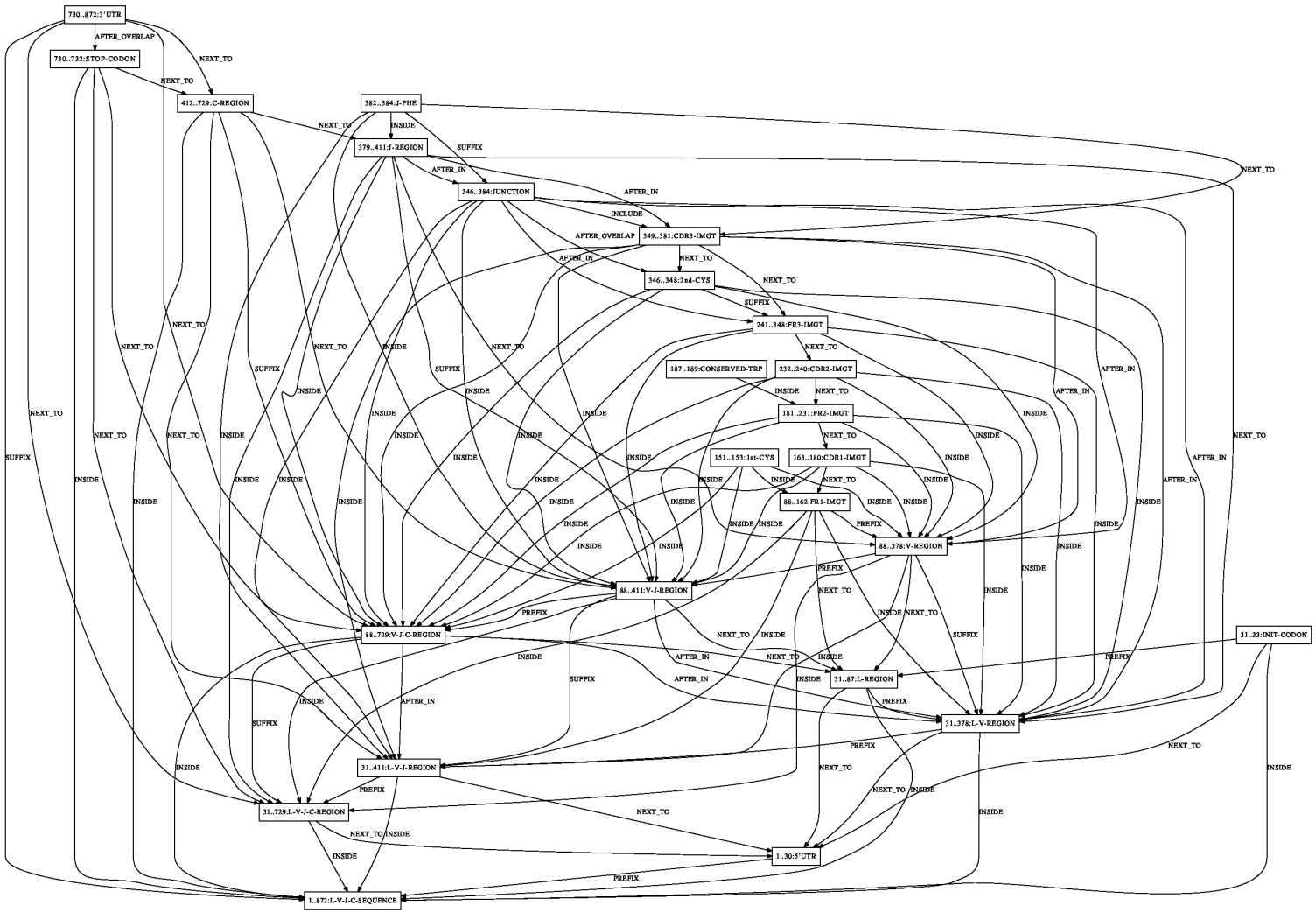
- retrouver les relations de l'ontologie IMGT®,
- comment considérer une relation comme :
  $[1;9] \star [10 = 9 + 1; 15]$
- solution ?



$[1;9] \mapsto [1;10[$

Exemple séquence M18645

Soit une annotation comportant 25 labels donc 25 intervales, $(25^2 - 25)/2 = 300$ comparaisons et donc $300 \times 4 = 1200$ équations dans le pire des cas.



Logiques de description

Introduction

- Représentation des connaissances
- Raisonnement : classification, inférence
- Formalisme logique : (méta-)propriétés
- Langages informatiques : DAML+OIL, Owl

Historique

- Langage de «frames»
- Réseaux sémantiques
- Graphes conceptuels
- Systèmes à base de connaissances : KL-ONE, BACK et LOOM
- Web sémantique et ontologie

dans le contexte de la logique formelle : propositionnelle et prédicative (longue histoire)

Syntaxe

La syntaxe des langages de description est donnée par 3 types : les individus I, les concepts C et les rôles R.
Il y a essentiellement 2 niveaux d'assertions :

- le général qui portent sur les concepts et les rôles,
- le factuel qui portent sur les individus.

Individus

Ce sont des identifiants, donc $\mathcal{I}$ est la donnée d'un ensemble de symbole ou atome. $I \equiv \quad \mathcal{I} \mid \quad$ atome
Exemple : Anne, Sophie, Robert, David

Rôles

$R \equiv$	$\mathcal{R}$		atome
	$\top_R$	anything	
	$\bot_R$	nothing	
	$R \sqcap R$	r-and	et
	R/C	range	restreint à
	R^{-1}	inverse	inverse de
	$R \circ R$	compose	composée de

Exemple : estParentDe

Concepts

$C \equiv$	$\mathcal{C}$		atome
	$\top$	anything	
	$\bot$	nothing	
	$\neg\mathcal{C}$	a-not	non atomique
	$\neg C$	not	non généralisée
	$C \sqcap C$	c-and	et
	$C \sqcup C$	c-or	ou
	$\forall R.C$	all	tous
	$\exists R$	some	quelque
	$\geq nR$	atleast	au moins
	$\leq nR$	atmost	au plus
	$\exists R.C$	c-some	
	$\geq nR.C$	c-atleast	au moins
	$\leq nR.C$	c-atmost	au plus
	$\{\mathcal{I}, \ldots, \mathcal{I}\}$	one-of	un parmis

Exemple : Male, Femelle, Humain, Homme, Femme, Pere, Mere

Propriétés

Quelques équivalences :

$$\begin{aligned} \exists R.\top &\equiv \exists R \\ \leq nR.\top &\equiv \leq R \\ \geq nR.\top &\equiv \geq R \end{aligned}$$

Terminologie $\mathcal{T}$

La $\mathcal{T} - Box$ est la boîte dite «terminologique». Elle détermine la structure conceptuelle à l'aide d'un ensemble de deux formes d'assertion :

- la *subsomption* $C \sqsubseteq C$,
- la *définition* $C \doteq C$, qui est en fait une double subsomption (symétrique).

La subsomption représente le fait que tous les individus d'un concept sont aussi des individus d'un autre concept. Autrement dit, son interprétation correspond à l'inclusion.
La définition représente le fait que tous les individus d'un concept sont exactement les individus d'un autre concept. Généralement l'un des deux membres de l'égalité est plus complexe que l'autre. Autrement dit, son interprétation correspond à l'égalité.

Assertion $\mathcal{A}$

La $\mathcal{A} - Box$ est la boîte dite «ontologique». Elle détermine la structure individuelle à l'aide de deux formes d'assertion :

- la *conception* $C(I)$,
- la *relation* $R(I, I)$.

Exemple

$\mathcal{T}$ − *Box*	$\mathcal{A}$ − *Box*
Femelle $\sqsubseteq \top \sqcap \neg$ Male	
Male $\sqsubseteq \top \sqcap \neg$ Femelle	Humain(Anne)
Animal $\doteq$ Male $\sqcup$ Femelle	Femelle(Anne)
Humain $\sqsubseteq$ Animal	Femme(Sophie)
Femme $\doteq$ Humain $\sqcap$ Femelle	Humain(Robert)
Homme $\doteq$ Humain $\sqcap \neg$ Femelle	$\neg$ Femelle(Robert)
Mere $\doteq$ Femme $\sqcap \exists$ estParentDe	Homme(David)
Pere $\doteq$ Homme $\sqcap \exists$ estParentDe	estParentDe(Sophie,Anne)
MereSansFille $\doteq$ Mere $\sqcap \forall$ estParentDe . $\neg$ Femme	estParentDe(Robert,David)
estParentDe $\sqsubseteq \top_R$	

Interprétation

Une interprétation est la donnée d'un domaine $\mathcal{D}$ et d'une fonction $\|-\| : \mathcal{L} \longrightarrow \mathcal{D}$, devant être définie pour chacun des trois types et de manière à respecter les assertions.

Les individus

L'interprétation de chaque individu est un élément du domaine, autrement dit $\|I\| \subseteq \mathcal{D}$. Les contextes complètent l'interprétation. Pour cela, on donne l'interprétation de chaque individu de la $\mathcal{A} - \mathit{Box}$.

Les concepts

L'interprétation d'un concept est une sous-ensemble du domaine.

$$
\begin{aligned}
\|\top\| &= \mathcal{D} \\
\|\bot\| &= \emptyset \\
\|\neg C\| &= \mathcal{D} \setminus \|C\| \\
\|C \sqcap C\| &= \|C\| \cap \|C\| \\
\|C \sqcup C\| &= \|C\| \cup \|C\| \\
\|\forall R.C\| &= \{d \in \mathcal{D} \mid \forall e : (d,e) \in \|R\| \Rightarrow e \in \|C\|\} \\
\|\exists R\| &= \{d \in \mathcal{D} \mid \exists e : (d,e) \in \|R\|\} \\
\|\geq n R\| &= \{d \in \mathcal{D} \mid \sharp\{e \mid (d,e) \in \|R\|\} \geq n\} \\
\|\leq n R\| &= \{d \in \mathcal{D} \mid \sharp\{e \mid (d,e) \in \|R\|\} \leq n\}
\end{aligned}
$$

Les rôles

L'interprétation d'un rôle est un sous-ensemble du produit cartésien sur le domaine.

$$
\begin{aligned}
\|R \sqcap R\| &= \|R\| \cap \|R\| \\
\|R/C\| &= \{(d,e) \in \|R\| \mid d,e \in \|C\|\} \\
\|R^{-1}\| &= \{(e,d) \mid (d,e) \in \|R\|\} \\
\|R \circ R\| &= \{(d,f) \mid (d,e) \in \|R\| \cup (e,f) \in \|R\|\}
\end{aligned}
$$

Subsomption

La subsomption est une contrainte forte car elle doit être respecter par toute interprétation $\|-\|$.

Inférence

Au niveau terminologique
Étant donnée une terminologie, les principaux problèmes d'inférence sont :

- satisfaction : un concept C est satisfaisable si et seulement si il existe un modèle $\|-\|$ tel que $\|C\| \neq \emptyset$
- subsomption : un concept C_1 est subsumé par un concept C_2 si et seulement si $\|C_1\| \subset \|C_2\|$ pour tout modèle $\|-\|$,
- équivalence : un concept C_1 est équivalent à un concept C_2 si et seulement si $\|C_1\| = \|C_2\|$ pour tout modèle $\|-\|$,
- disjonction : deux concepts C_1 et C_2 sont disjoints si et seulement si $\|C_1\| \cap \|C_2\| = \emptyset$ pour tout modèle $\|-\|$.

Inférence (2)

Au niveau factuel

Étant données une terminologie $\mathcal{T}$ et une ontologie $\mathcal{A}$:

- cohérence : $\mathcal{A}$ est cohérente par rapport à $\mathcal{T}$ si et seulement si il existe un modèle,
- vérification d'instance : vérifier par inférence si une assertion $C(I)$ est vraie pour tout modèle,
- vérification de rôle : vérifier par inférence si une assertion $R(I, I)$ est vraie pour tout modèle,
- extraction : inférer les individus d'un concept pour tout modèle.

Interroger le Web

Quelques repères

- ARPANET (1969-1972) => TCP/IP (1974) => Internet (1980)
- Hypertext (Ted Nelson, 1965) => WWW (Tim Berners-Lee, 1989) => HTTP 0.9 (1991)
- Navigateur HyperCard (1987) => ViolaWWW (1992) => NCSA Mosaic (fin 1992-1993) : 300 sites dispo ! ! ! !
- aujourd'hui Web = immense source d'information ?
- information = données ?
- immense, mouvante, «informe»

Web pour les humains

Moteurs de recherche

- trouver l'information la plus pertinente
- indexer et estimer des billions de «pages» web (adresse)
- déterminer ce qui est signifié

mais seulement pour...

- indiquer des pages pouvant contenir un élément de réponse
- confier l'interprétation à l'utilisateur

Au delà de la navigation

- information exploitable par logiciels
- langage de requêtes et de transformation
- intégration en document multimédia
- coordonnées web : Uniform Resource Locator

Web pour les programmes

- Formats : XML, RIF, RDF, représentation commune
- Sémantique : Ontologie d'un tout ? vision dynamique (extensible et réutilisable), mettre en correspondance (mapping) et raisonner
- E/S humaine : TALN, visualisation, interface utilisateur, XSLT
- Méthode d'accès : résolution d'URI, (semantic web) services, REST, SPARQL

Adresse Web

- Uniform Resource Locator (URL)

  `http://www.imgt.org/IMGTrepertoire/Proteins/proteinDisplays.php?species=human&group=IGH`
  `ftp://ftp.cines.fr/IMGT/`
  `mailto:firstname.lastname@whatever.net`

- Uniform Resource Name (URN)

  `urn:isbn:0-395-36341-1`
  `urn:ietf:rfc:2141`

- URL + URN => Uniform Resource Identifier (URI)
- URI + caractères Unicode => Internationalized Resource Identifier (IRI)

RDF - Resource Description Framework

- Modèle simple de données à base de triplets :
  - des *ressources* (ayant un identifiant unique)
  - des *littéraux* (valeur simple)
  - des *relations* nommées entre deux ressources (ou une ressource et un littéral)
- pour permettre de :
  - décrire formellement la «sémantique» de l'information de manière accessible aux machines (programmes)
  - représenter des méta-données (données sur les données)
- Plusieurs représentations syntaxiques (N-Triples, RDF/XML, RDF/JSON, Turtle, ...)

RDF est un format de données façon pour «lier» des données dans le Web

RDF - Modèle de données

Une collection (liste) de triplets représentant une association (Sujet Prédicat Objet) où :

- un sujet est soit une IRI ou un noeud anonyme,
- un prédicat est une IRI,
- un objet peut être soit une IRI, soit un noeud anonyme ou soit un littéral

triplestore = système de bases de triplets RDF

RDF - Types de données

- des IRI : une déclinaison internationale des URI

```
<http://example.org/path/#fragment>
</path>
foo:bar
```

- des littéraux : chaînes de caractères ou nombres

```
"bonjour" "chat"@en "chat"@fr "foo"^^<http://example.org/my/datatype>
"10"^^xsd:decimal
"""ici il y a des
retour à la ligne
"""
```

- des noeuds anonymes

```
_:truc _:
[]
```

RDF - Exemple 1 (RDF/XML)

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description rdf:about="http://fr.wikipedia.org/wiki/Lewis_Carroll">
          <dc:title>Lewis Carroll</dc:title>
          <dc:publisher>Wikipedia</dc:publisher>
          <foaf:primaryTopic>
              <foaf:Person>
                  <foaf:name>Charles Lutwidge Dodgson</foaf:name>
              </foaf:Person>
          </foaf:primaryTopic>
      </rdf:Description>
</rdf:RDF>
```

RDF - Exemple 1 (Turtle)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://fr.wikipedia.org/wiki/Lewis_Carroll> dc:title "Lewis Carroll" ;
        dc:publisher "Wikipedia" ;
        foaf:primaryTopic _:node16prp334px64161653 .

_:node16prp334px64161653 a foaf:Person ;
        foaf:name "Charles Lutwidge Dodgson" .

<http://any23.org/tmp/> <http://vocab.sindice.net/date> "2012-05-20T18:51:17+01:00" ;
        <http://vocab.sindice.net/size> "7"^^<http://www.w3.org/2001/XMLSchema#int> .
```

Quelques schémas RDF

- RDF Schema (RDFS) : schéma des concepts RDF
- Dublin Core (DC) : schéma de métadonnées pour les documents numériques
- Friend of a friend (FOAF) : schéma réseaux sociaux
- Exif RDF : schéma de métadonnées image et photo numérique

RDFS - RDF Schema

- langage de description de vocabulaire (méta-langage)
- standard W3C, version 1.0 (2004)
- des types : `rdfs:Resource` et `rdfs:Class`.
- des propriétés : `rdf:type` (noté aussi `a`) et `rdfs:subClassOf`.
- `rdfs:domain` et `rdfs:range`.

RDFS - Des classes

Element	classe des	rdf:type	rdfs:subClassOf
rdf:Statement	déclarations	rdfs:Class	rdfs:Resource
rdf:Property	propriétés	rdfs:Class	rdfs:Resource
rdf:XMLLiteral	valeurs litérales XML	rdfs:Datatype	rdfs:Literal
rdf:List	listes	rdfs:Class	rdfs:Resource
rdf:Bag	collections non-ordonnées	rdfs:Class	rdfs:Container
rdf:Seq	collections ordonnées	rdfs:Class	rdfs:Container
rdf:Alt	collections d'alternatives	rdfs:Class	rdfs:Container
rdfs:Resource	ressources	rdfs:Class	rdfs:Resource
rdfs:Class	classes	rdfs:Class	rdfs:Resource
rdfs:Literal	valeurs litérales	rdfs:Class	rdfs:Resource
rdfs:Datatype	types de données	rdfs:Class	rdfs:Class
rdfs:Container	collections	rdfs:Class	rdfs:Resource
rdfs:ContainerMembershipProperty	propriétés d'appartenance	rdfs:Class	rdf:Property

RDFS - Des propriétés

Element	Sémantique	rdfs:domain	rdfs:range
rdf:type	instance de	rdfs:Resource	rdfs:Class
rdf:value	for structured values	rdfs:Resource	rdfs:Resource
rdf:subject	sujet de la déclaration	rdf:Statement	rdfs:Resource
rdf:predicate	prédicat de la déclaration	rdf:Statement	rdfs:Resource
rdf:object	objet de la déclaration	rdf:Statement	rdfs:Resource
rdf:first	tête de la liste	rdf:List	rdfs:Resource
rdf:rest	queue de la liste	rdf:List	rdf:List
rdf:_1, rdf:_2, ...	élément d'un conteneur	rdfs:Container	rdfs:Resource
rdfs:range	sujets restreint à	rdf:Property	rdfs:Class
rdfs:domain	objets restreint à	rdf:Property	rdfs:Class
rdfs:subClassOf	sous-classe de	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	sous-propriété de	rdf:Property	rdf:Property
rdfs:member	appartient à	rdfs:Resource	rdfs:Resource
rdfs:label	label (humain)	rdfs:Resource	rdfs:Literal
rdfs:comment	commentaire (humain)	rdfs:Resource	rdfs:Literal
rdfs:seeAlso	autre information	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	définition	rdfs:Resource	rdfs:Resource

RDFS - Extrait du RDF/XML

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">

 <owl:Ontology
      rdf:about="http://www.w3.org/2000/01/rdf-schema#"
      dc:title="The RDF Schema vocabulary (RDFS)"/>

<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Resource</rdfs:label>
  <rdfs:comment>The class resource, everything.</rdfs:comment>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Class">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Class</rdfs:label>
  <rdfs:comment>The class of classes.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>
```

...

```
<rdf:Description rdf:about="http://www.w3.org/2000/01/rdf-schema#">
  <rdfs:seeAlso rdf:resource="http://www.w3.org/2000/01/rdf-schema-more"/>
</rdf:Description>

</rdf:RDF>
```

RDFS - Extrait du Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.w3.org/2000/01/rdf-schema#> a owl:Ontology ;
        dc:title "The RDF Schema vocabulary (RDFS)" .

rdfs:Resource a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Resource" ;
        rdfs:comment "The class resource, everything." .

rdfs:Class a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Class" ;
        rdfs:comment "The class of classes." ;
        rdfs:subClassOf rdfs:Resource .
```

...

```
<http://www.w3.org/2000/01/rdf-schema#> rdfs:seeAlso <http://www.w3.org/2000/01/rdf-schema-mo

<http://www.w3.org/2000/01/rdf-schema> <http://vocab.sindice.net/date> "2012-05-19T17:29:53+0
        <http://vocab.sindice.net/size> "89"^^<http://www.w3.org/2001/XMLSchema#int> .
```

SPARQL - SPARQL Protocol and RDF Query Language

- tentative antérieure avec entre autres RDQL (W3C 2004)
- standard W3C, version 1.0 (2008) et bientôt 1.1
- obtenir des valeurs de données structurées ou semi-structurées
- explorer les données en des relations avec inconnues
- réaliser des jointures complexes de bases disparates en une seule et simple requête
- transformer des données RDF d'un vocabulaire à un autre

SPARQL - Schématiquement

Une requête SPARQL comprend dans l'ordre :

1. déclaration de préfixes pour simplifier l'écriture des IRI
2. spécification des données d'entrée
3. spécification des données de sortie (variables ou litéraux)
4. le motif de la recherche, spécifiant ce qui est recherché dans les données d'entrée
5. des modificateurs tels que le partionnement, l'ordonnancement et autres concernant le résultat

SPARQL - Requête

```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...
# result
                          clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

Requête simple

- La base :

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "Ceci n'est pas
```

- La requête :

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

- Le résultat ?

Requête à réponse multiple

- La base :

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Gaston Lagaffe" .
_:a  foaf:mbox   <mailto:glagaffe@example.com> .
_:b  foaf:name   "Tintin" .
_:b  foaf:mbox   <mailto:tintin@example.org> .
_:c  foaf:mbox   <mailto:snoopy@example.org> .
```

- La requête :

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

- Le résultat ?

Requête avec filtre

- La base :

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix : <http://example.org/book/>.

:book1 dc:title "Ceci n'est pas un livre".
:book2 dc:title "Tout un programme".
```

- La requête :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE {
        ?x dc:title ?title .
        FILTER regex(?title, "^Ceci")
}
```

- Le résultat ?

Meta-propriétés (1)

- L'héritage de type par `rdfs:subclassOf`. Par exemple, les déclarations

  ```
  (Milou rdf:type Chien)
  (Chien rdfs:subClassOf Mammifère)
  ```

  implique le fait

  ```
  (Milou rdf:type Mammifère)
  ```

- La réflexivité de `rdfs:subPropertyOf` et `rdfs:subClassOf`. Ainsi pour tout `rdf:Property` *p*, le fait :

  ```
  (p rdfs:subPropertyOf p)
  ```

  est déduit. De même pour toute `rdfs:Class` *C*, le fait :

  ```
  (C rdfs:subClassOf C)
  ```

  is inferred.

Meta-propriétés (2)

- L'inférence de type par `rdfs:range` et `rdfs:domain`. Par exemple, les faits :

  ```
  (enseigne rdfs:domain Enseignant)
  (enseigne rdfs:range Etudiant)
  (Platon enseigne Aristote)
  ```

  implique les faits :

  ```
  (Platon rdf:type Enseignant)
  (Aristote rdf:type Etudiant)
  ```

- La transitivité de `rdfs:subClassOf` et de `rdfs:subPropertyOf`. Par exemple, si nous avons les faits :

  ```
  (Chien rdfs:subClassOf Mammifère)
  (Mammifère rdfs:subClassOf Animal)
  ```

  alors nous avons aussi le fait :

  ```
  (Chien rdfs:subClassOf Animal)
  ```

  Similairement, les faits :

  ```
  (parent rdfs:subPropertyOf ancestor)
  (ancestor rdfs:subPropertyOf relative)
  ```

RDF et raisonner

- déduire des faits (implicites)
- minimiser les faits (déclarés)
- intension de «méta-propriétés»
- plus généralement : règles logiques

=> Rule Interchange Format (RIF)

XML - eXtensible Markup Language

- Script => GML (1967) => SGML (ISO 8879 :1986) => XML (1997), format ouvert, standardisé W3C, deux versions 1.0 ou 1.1
- syntaxe par système de balises (simple, générique et extensible)
- balises définies par DTD ou XML Schema (2001)
- espaces de noms (plusieurs systèmes de balises)
- arborescence d'éléments avec propriétés

XML - Exemple XHTML+SVG

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Inclusion de SVG dans du XHTML</title>
</head>
<body>

<h1>Du SVG dans du XHTML</h1>

<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
        version="1.1" width="400px" height="300px">

<defs>
<linearGradient id="deg" x1="0" y1="0" x2="1" y2="1">
        <stop offset="0%" style="stop-color:skyblue;stop-opacity:0"/>
        <stop offset="100%" style="stop-color:skyblue;stop-opacity:0.8"/>
</linearGradient>
</defs>

<rect x="10" width="380" y="10" height="280" style="fill:url(#deg)"/>
<text x="30" y="30"
        style="font-weight:bold;font-size:30px"
        dy="0 8 16 24 32 25 25 0 80">
Texte en
<tspan style="font-size:100px" rotate="-10 -2 15">SVG</tspan>
</text>

</svg>
</body>
</html>
```

XQuery - XML Query language

- basé sur XPath (deux versions 1.0 et 2.0)
- mais n'est pas du XML !
- approche différente à XSLT (transformation XML à XML)

XQuery Data Model

- itérer une liste (séquence finie) d'entités
- et donner une liste d'entités
- où une entité peut être :
  - une valeur élémentaire (numérique ou textuelle),
  - un élément XML.

XSPARQL

- un langage de requête commun pour RDF et XML, combinaison de XQuery et SPARQL
- consomme et génère du XML ou RDF
- langage de transformation
- extension syntaxique à XQuery

XSPARQL - Schéma syntaxe principal (I)

⟨*P*⟩ = `declare namespace` ⟨*prefix*⟩ = " ⟨*namespace-URI*⟩ "
| `prefix` ⟨*id*⟩ : < ⟨*namespace-URI*⟩ >

⟨*F*⟩ = `for` ⟨*var*⟩ [`at` ⟨*posVar*⟩] `in` ⟨*expr*⟩

⟨*F'*⟩ = `for` ⟨*var*⟩ [`at` ⟨*posVar*⟩]

⟨*D*⟩ = `from` [`named`] (⟨*dataset-URI*⟩
| ⟨*expr*⟩)

⟨*L*⟩ = `let` ⟨*var*⟩ := ⟨*expr*⟩

⟨*W*⟩ = `where` ⟨*expr*⟩

⟨*O*⟩ = `order by` ⟨*expr*⟩

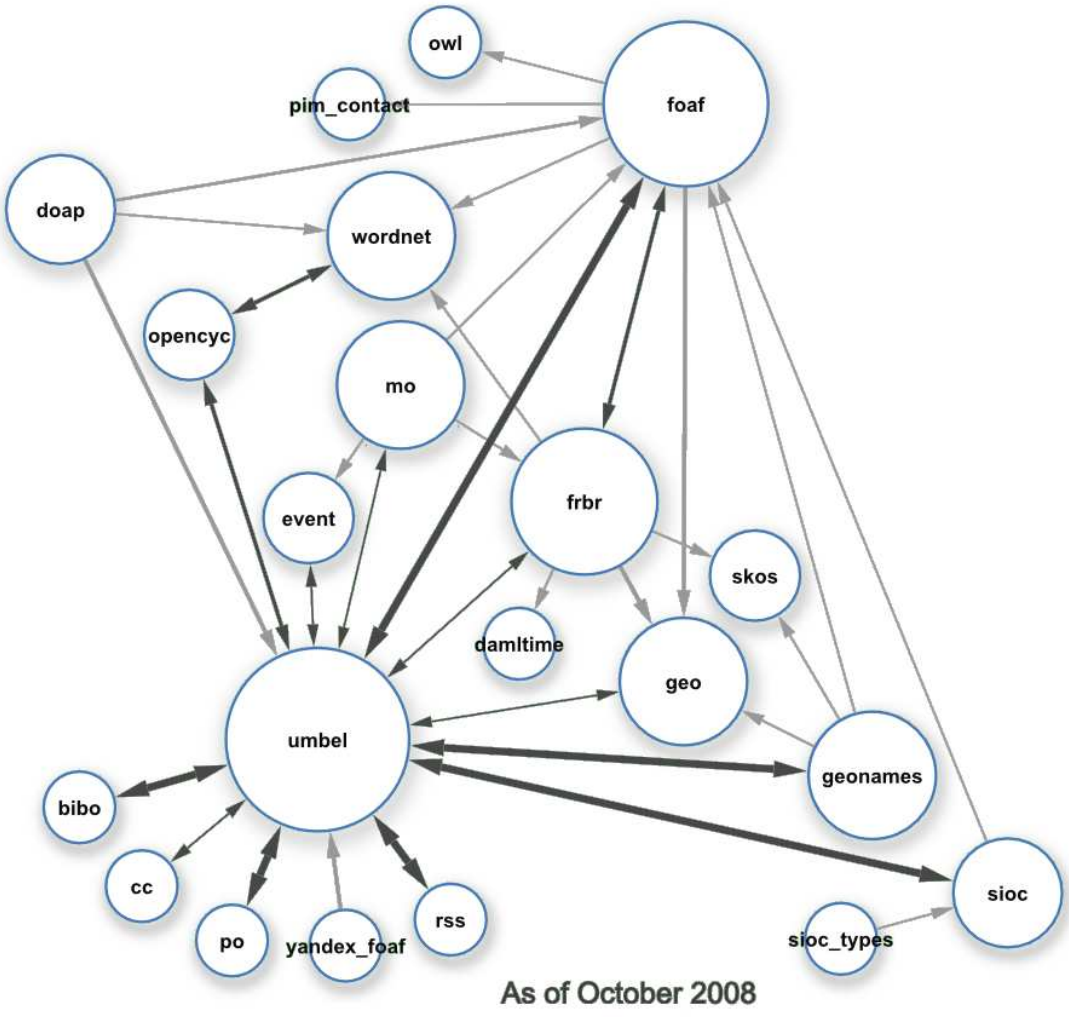
⟨*M*⟩ = `limit` ⟨*posinteger*⟩

⟨*M*⟩ = `offset` ⟨*posinteger*⟩

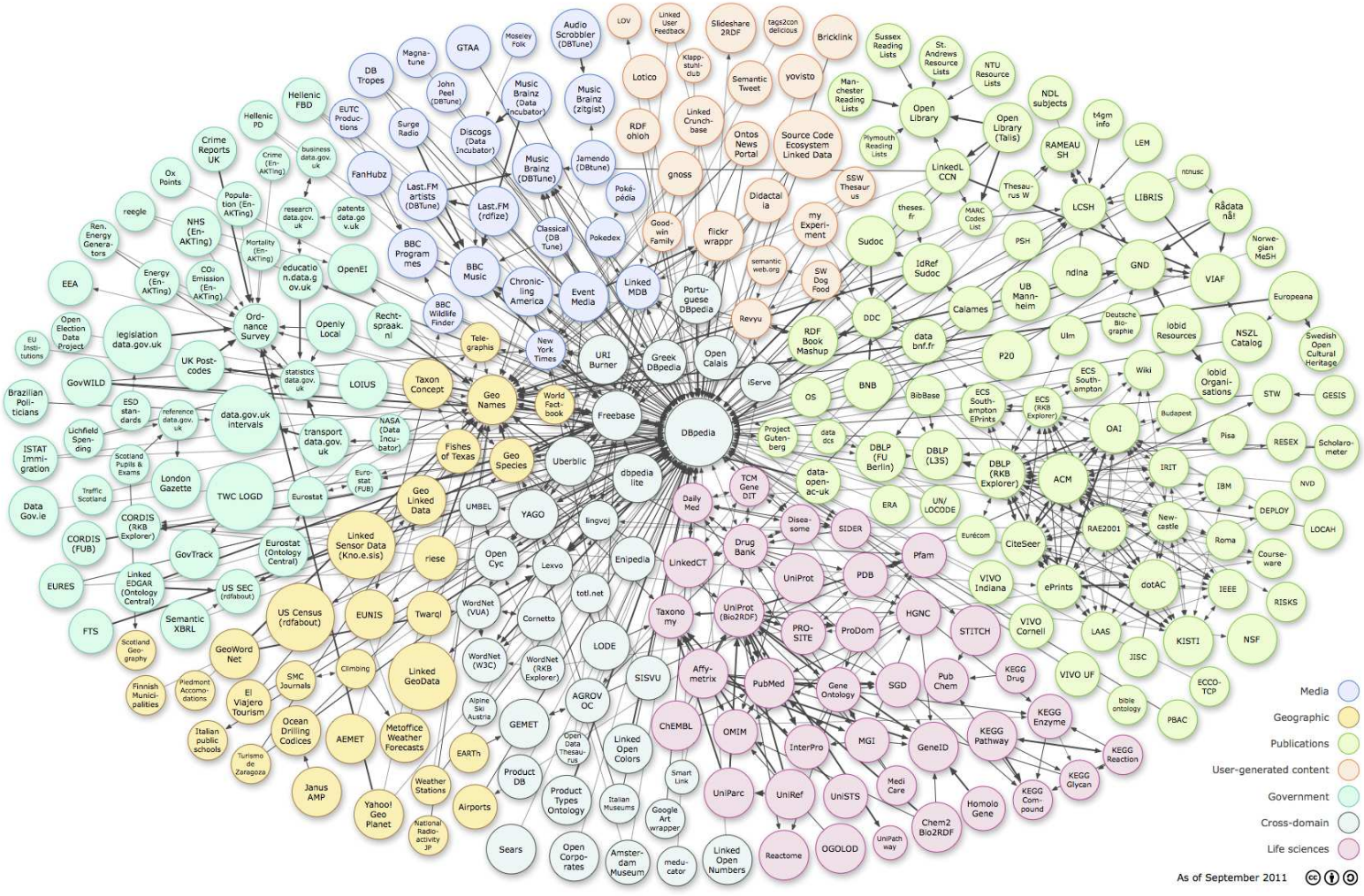
⟨*C*⟩ = `construct {` ⟨*template*⟩ `}`

⟨*R*⟩ = `return` ⟨*XML*⟩

Aperçu en octobre 2008



Aperçu en septembre 2011



SPARQL - vers la version 1.1

- les agrégats (COUNT, HAVING)
- les sous-requêtes (SELECT)
- les négations (MINUS et FILTER NOT EXISTS)
- les expressions dans la projection (AS)
- les chemins de propriété (expressions régulières d'IRI)
- les mises à jour : sur une base (CREATE, DROP, LOAD, CLEAR) sur un triplet (INSERT, DELETE)
- les inférences...

Jouons un peu !

- SPARQL 1.1 Query Language : `http://www.w3.org/TR/sparql11-query/`
- DBpedia : `http://dbpedia.org/`
- SPARQL Endpoints : `http://www.w3.org/wiki/SparqlEndpoints`
- `http://www.sparql.pro/`
- Visual RDF : `http://graves.cl/visualRDF/`
- Welkin : `http://simile.mit.edu/welkin/`
- Javascript SPARQL result set visualizer : `http://code.google.com/p/sgvizler/`

Pour aller plus loin...

- XSPARQL Use Cases : `http://www.w3.org/Submission/xsparql-use-cases/`
- XSPARQL Bridging the RDF and XML worlds : `http://xsparql.deri.org/`
- DERI Pipes : Open Source, Extendable, Embeddable Web Data Mashups : `http://pipes.deri.org/`
- Accessing Relational Databases as Virtual RDF Graphs : `http://d2rq.org/`