NoSQL - Systèmes de gestion de données distribués

I. Mougenot isabelle.mougenot@umontpellier.fr

Faculté des Sciences Université Montpellier

2015

Quand passer par un système NoSQL ?

Alternatives au relationnel dans des cas de figure ciblés

- ”plasticité” du schéma & gros volumes de données distribuées
- entités protéiformes : nombreuses caractéristiques +/- renseignées
- de multiples associations entre entités avec multiplicités 0..*
- des attributs multivalués et composites

Problème posé par les entités ”protéiformes”

Premier effort de modélisation pour mieux comprendre le problème posé

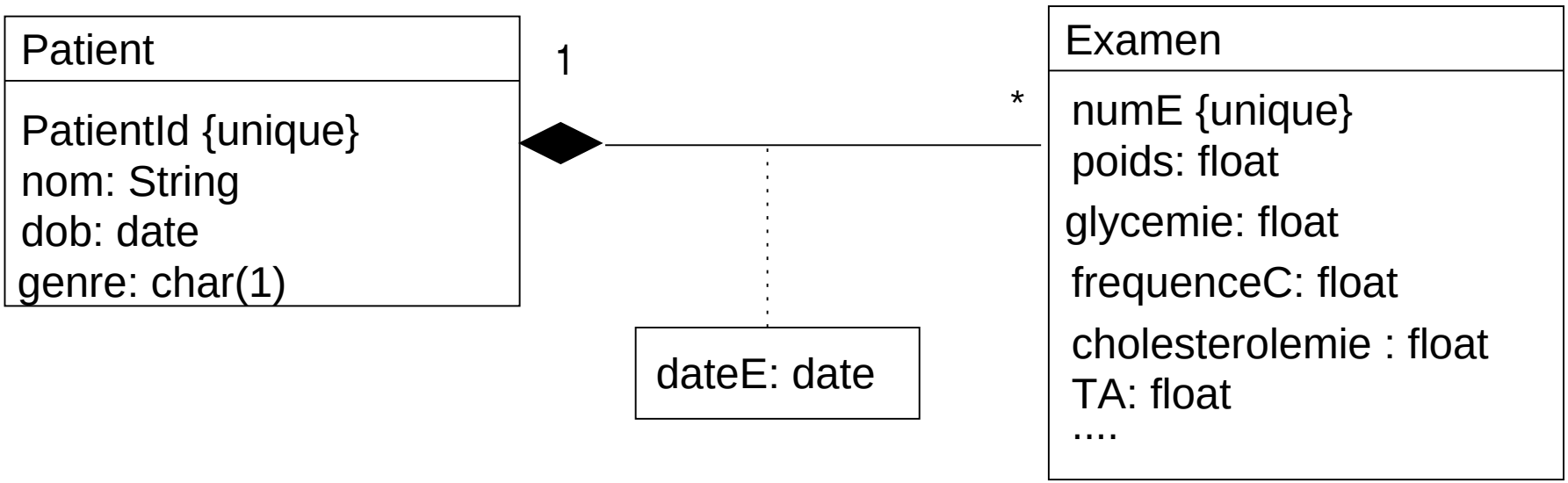

Figure: Diagramme UML : dossier patient

Modèle relationnel : traduction du modèle UML précédent

Patient(**PatientId**, nom, dob, genre)

Examen(**NumE**, dateE, PatientId, Poids, Glycemie, FrequenceC, ...)
avec PatientId $\subseteq$ Patient(PatientId)

Premier problème : révision fréquente du schéma et notamment des colonnes des tables

Problème posé par les entités ”protéiformes”

Relation Examen en extension

numE	dateE	PatientId	Poids	Glycemie	FrequenceC	...
1	20/10/10	3	79	null	null	...
2	28/10/10	3	null	6	null	...
3	28/10/10	2	null	null	170	...

Figure: Schéma relationnel inapproprié

Modèle Entité-Attribut-Valeur (EAV)

Rendre une partie du schéma générique vu parfois comme un anti-patron en BD

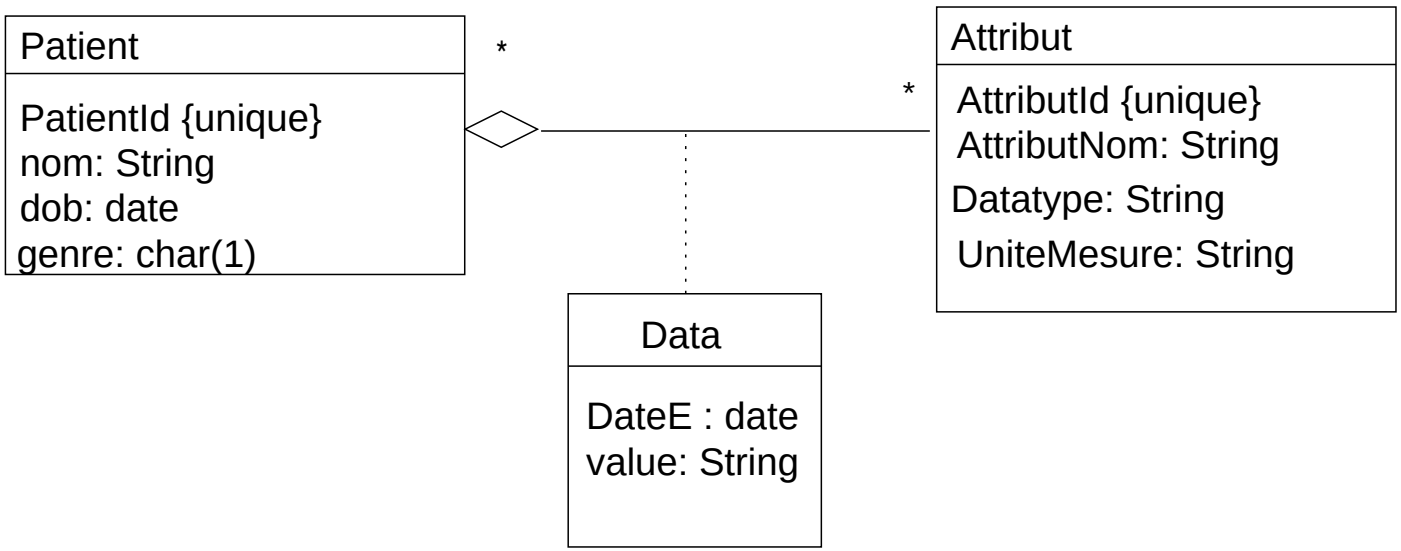

Figure: Diagramme de classes révisé

Schéma relationnel

Patient(**PatientId**, nom, dob, genre)

Attribut(**AttributId**, AttributNom, Datatype, UniteMesure)

Data (**PatientId, AttributId, dateE**, value)
avec PatientId $\subseteq$ Patient(PatientId) et AttributId $\subseteq$ Attribut(AttributId)

Illustration relations en extension

PatientId	Nom	Dob	genre
3	Pierre Martin	20/10/1970	M
2	Marie Monin	20/10/1987	F

Figure: Extension possible Patient

AttributId	AttributNom	DataType	UniteMesure
1	Poids	float	Kg
2	FrequenceC	float	puls/min
3	Glycemie	float	mmole/l
4	Diagnostic	string	null

Figure: Extension possible Attribut

Illustration relations en extension

Relation Data en extension

AttributId	dateE	PatientId	Value
1	20/10/10	3	79
3	28/10/10	3	6
2	28/10/10	2	170

Figure: Extension possible Data

EAV : approche hybride

Distinction de différents niveaux dans le modèle

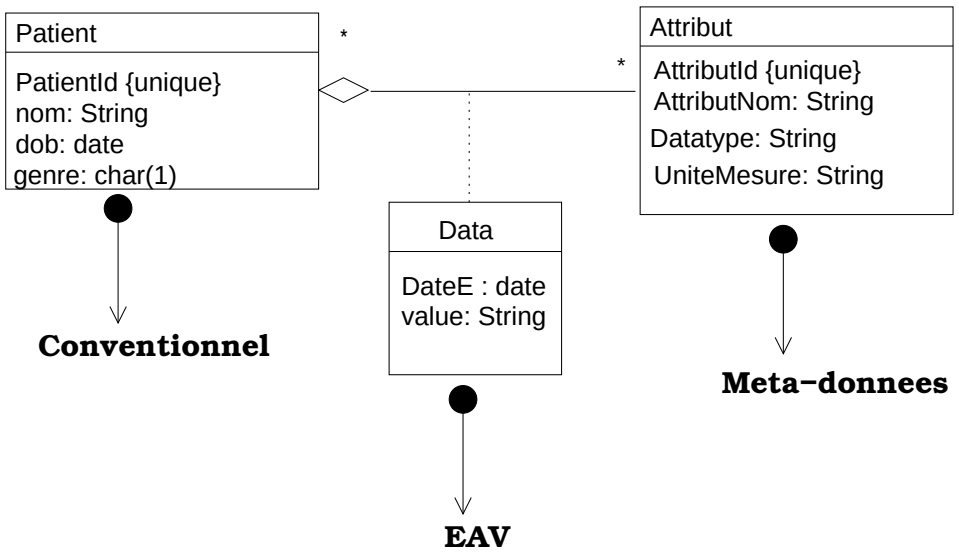

Figure: Diagramme de classes révisé

Dans le monde de la BD relationnelle

Gérer trois niveaux d'information au sein même de la base de données

- Dictionnaire de données ou méta-schéma : table des tables, table des attributs dans notre cas, la relation Attribut du modèle peut être vue à ce niveau, table des contraintes, . . .
- Schéma de données : table Patient, table Medecin, . . .
- Monde réel : tuples ou faits de la BD : patients Pierre Martin, Marie Monin, . . .

UNIVERSITÉ DE MONTPELLIER

Prendre en charge des attributs de type LOB

Les données de type image : rayon X, échographie, IRM, . . . nécessitent des considérations complémentaires

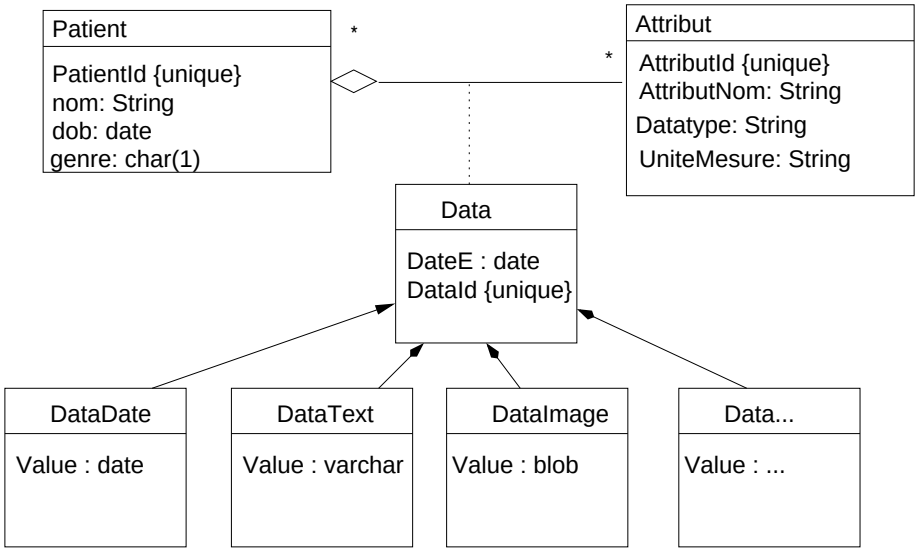

Figure: De nouvelles classes pour une meilleure prise en charge des types de données associées aux valeurs des attributs

Illustration relations en extension

AttributId	dateE	PatientId	DataId
1	20/10/10	3	1
3	28/10/10	3	2
2	28/10/10	2	3

Figure: Extension possible Data

dataId	Value
1	79
2	6
3	170

Figure: Extension possible DataFloat

Modèles relationnels inadaptés

Une solution générique, flexible et efficace (en matière de stockage) mais qui pose quelques problèmes à :

- l'alimentation de la base de données (éclatement de l'information, nécessité d'étendre les tests de vérification de la validité des données saisies),
- l'interrogation (requêtes SQL rendues plus complexes),
- l'affichage (fournir une information complète et adaptée à la perception de l'usager final).

Exemple de requête simple en algèbre relationnelle

Donner les patients qui pèsent plus de 70 kgs et qui ont une fréquence de moins de 120 puls/mn

Modèle de départ : Π PatientId, nom, dob, genre (œ poids $>$ 70 $\wedge$ frequenceC $<$120 (Patient $\bowtie$ Examen))

Modèle EAV simple :
Π PatientId, nom, dob, genre (œ AttributNom='Poids'$\wedge$ Value $>$ 70 (Patient $\bowtie$ Data $\bowtie$ Attribut))
$\cap$
Π PatientId, nom, dob, genre (œ AttributNom='FrequenceC' $\wedge$ Value $<$120 (Patient $\bowtie$ Data $\bowtie$ Attribut))

EAV/CR

CR pour Classes et Relations : ajout d'une couche objet pour plus d'expressivité dans le modèle ainsi que la prise en charge de structures complexes, notamment en ce qui concerne les métadonnées

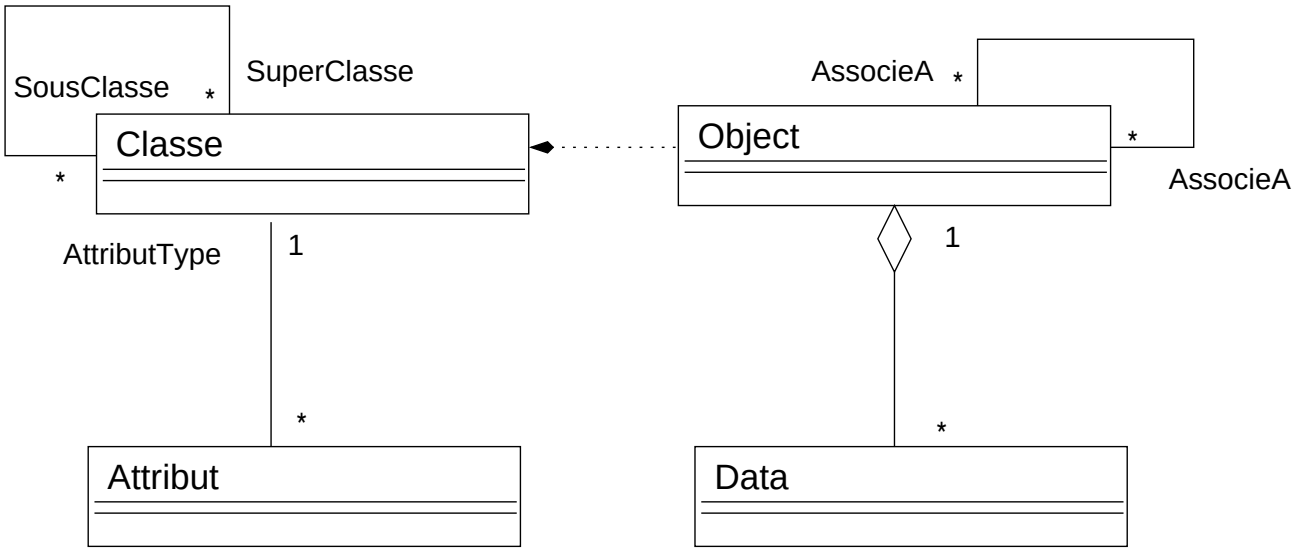

Figure: Diagramme de classes portant sur les aspects CR

Intérêts

Maîtriser le requêtage et l'affichage

- Un objet est vu comme une collection de données qui vont pouvoir être restituées de concert à l'usager, son voisinage avec d'autres objets va pouvoir également être exploité en terme d'affichage
- Une classe va permettre de typer les attributs et peut de plus être recadrée au travers d'une hiérarchie de classes : cet ajout de structuration peut être profitable pour une meilleure consultation de l'information (faciliter le requêtage)

Un tel dispositif peut être mis en oeuvre au travers des surcouches "objet/relationnel" des SGBDR et notamment de la notion de ADT

UNIVERSITÉ DE MONTPELLIER

Conclusions EAV

Le modèle EAV souvent décrié, souligne le besoin de schémas de données ouverts

- métamodèle RDF (triplet SPO) est en droite ligne de ce modèle
- systèmes NoSQL et la notion clé/valeur reprennent ce principe d'ouverture

Objectif : gérer de manière unifiée des entités munies de caractéristiques hétérogènes

Colonne et NF2 : principes des années 90 pour dépasser les limites du relationnel

Données complexes, multivaluées et éparses

- **modèle orienté colonne** : les colonnes deviennent les lignes : ”wide and sparse data” : faciliter l'évolution du schéma
- **modèle NF2** Non First Normal Form : s'affranchir de la contrainte de la première forme normale et gérer des données multi-valuées (collections) et composites (types complexes)

Systèmes orientés colonne

PatientCode	Firstname	LastName	DOB	Address
B_A_0001	Jean	Martin	5/5/55	Montpellier
B_A_0002	Marie	Dupont	6/6/66	Ales
B_A_0003	Marc	Dulac	7/7/77	Sete
...	...	...	...	...

PatientCode	B_A_0001	B_A_0002	B_A_0003	...
Firstname	Jean	Marie	Marc	...
LastName	Martin	Dupont	Dulac	...
DOB	5/5/55	6/6/66	7/7/77	...
Address	Montpellier	Ales	Sete	...

Figure: Rotation de 90 degrés

Systèmes orientés colonne : partitionnement vertical

∼ ”schema Aware” des triplestores

PatientCode	Firstname
B_A_0001	Jean
B_A_0002	Marie
B_A_0003	Marc
...	...

PatientCode	Address
B_A_0001	Montpellier
B_A_0002	Ales
B_A_0003	Sete
...	...

PatientCode	LastName
B_A_0001	Martin
B_A_0002	Dupont
B_A_0003	Dulac
...	...

Colonnes : quelques exemples de systèmes

A l'origine dans les années 80-90 : Decomposition Storage Model. SBGDR sous-jacents (usage de vues matérialisées avec partitionnement vertical ou définition d'index pour chaque colonne)

- **Académiques**
  - **C-Store (MIT Cambridge MA)** : depuis 2005
  - **MonetDB (CWI Amsterdam)** : pionnier en la matière
- **Commerciaux**
  - **Vertica (C-Store)**
  - **Sybase IQ**
  - **InfoBright MySQL**

Domaines d'application phare pour les systèmes orientés colonnes

Exploités ou source d'inspiration pour :

- Entrepôts de données
- Fouille de données
- Jeux de données scientifiques : santé, écologie, astrophysique, génomique fonctionnelle ...
- Modèles RDF (triple stores)
- **Google Big Table**

Modèle NF2 : inspiration TAD (SQL3)

Données composites et multi-valuées (collection)

PatientCode	Firstname	LastName	DOB	Symptoms		
				name	value	date
B_A_0001	Jean	Martin	5/5/55	sneezing	severe	3/3/3
	Ernest			itchy_troat	severe	3/3/3
B_A_0002	Marie	Dupont	6/6/66	...		
B_A_0003	Marc	Dulac	7/7/77	...		
	Antoine					
...	...	...	...	...		

Figure: Illustration modèle NF2

Modèle NF2 : SGBD Objet-Relationnel

Exemples de types de données abstraits avec Oracle

```
Create type Tsymptom as object
(name varchar(15), value varchar(10)
diag_date date, )
/
Create type coll_symptoms as Table of Tsymptom
/
Create table Patient (code varchar(8),
firstname coll_fsn, lastname varchar(20),
dob Date,
symptoms coll_symptoms);
Nested table symptoms Store As allSymptoms
```