SAADA

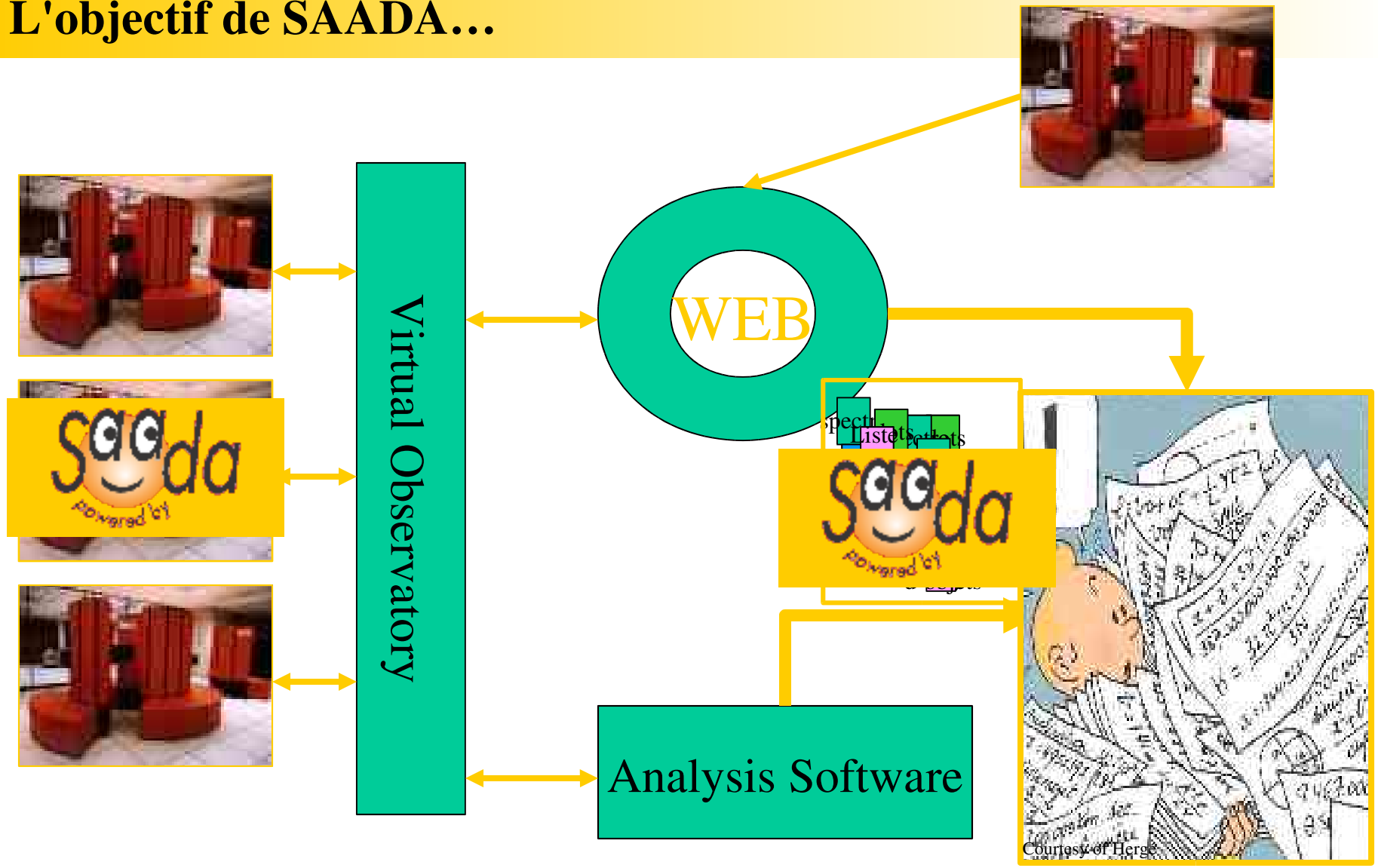
Fabriquer une base De données Astronomique Sans programmer

Team: Patrick MILLAN
Laurent MICHEL
Christian MOTCH
Ngoc Hoan NGUYEN
F. Xavier PINEAU

Funded by cnes Région Alsace

Supported by XMM-Newton SURVEY SCIENCE CENTRE

L'objectif de SAADA…



… Suite

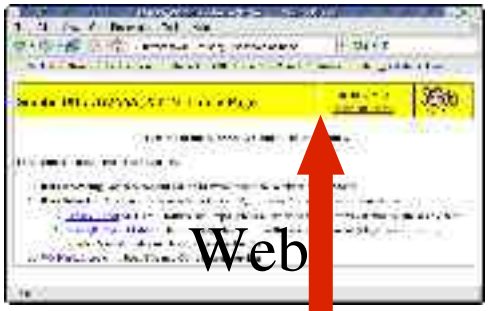
- **SAADA se propose de donner aux astronomes le moyen de mettre leurs données individuelles dans de vraies bases de données: les SaadaDBs.**
  - Les SaadaDBs sont conçues pour les données astronomiques
  - Les SaadaDBs sont dédiées aux astronomes
  - Les SaadaDBs sont auto-configurables – Pas de code à écrire -

- **SAADA est un outil pour faire de la Science.**
  - Des données sont archivées et sélectionnées en suivant des considérations scientifiques.
  - Les données peuvent être associées entre elles par des liens permanents.
  - Les données peuvent être manipulées par une API Java.
- **Saada est un outil pour exposer ses données.**
  - Interface Web
  - Publication dans l'Observatoire Virtuel

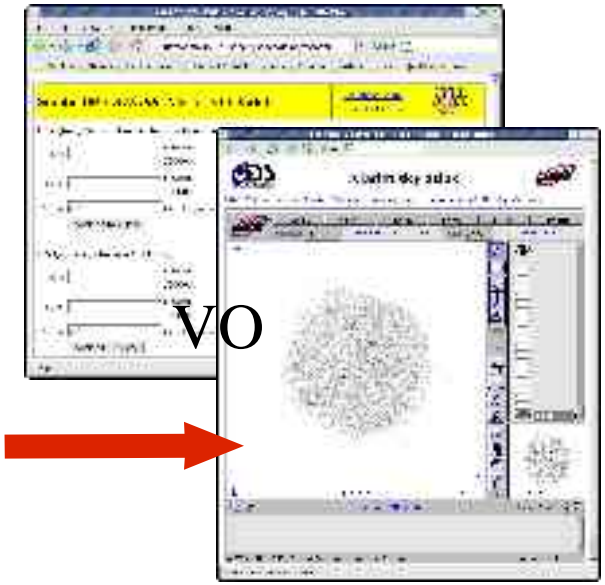
Le principe de base: Un générateur de base de données

- **Saada est un générateur de base de données**
  - Saada crée une base de données (une SaadaDB) vide suivant un canevas prédéfini
  - Une fois la SaadaDB créée, Saada ne sert plus à rien sinon à créer une autre SaadaDB.
- **La SaadaDB est autonome**
  - Elle possède ses propres outils.
  - Elle possède ses propres bibliothèques.
  - Elle possède sa propres base SQL
  - Elle possède sa propre URL d'accès

Principe de création d'une SaadaDB



Web



VO

API Java

SaadaDB opérationnelle

Règles de configuration

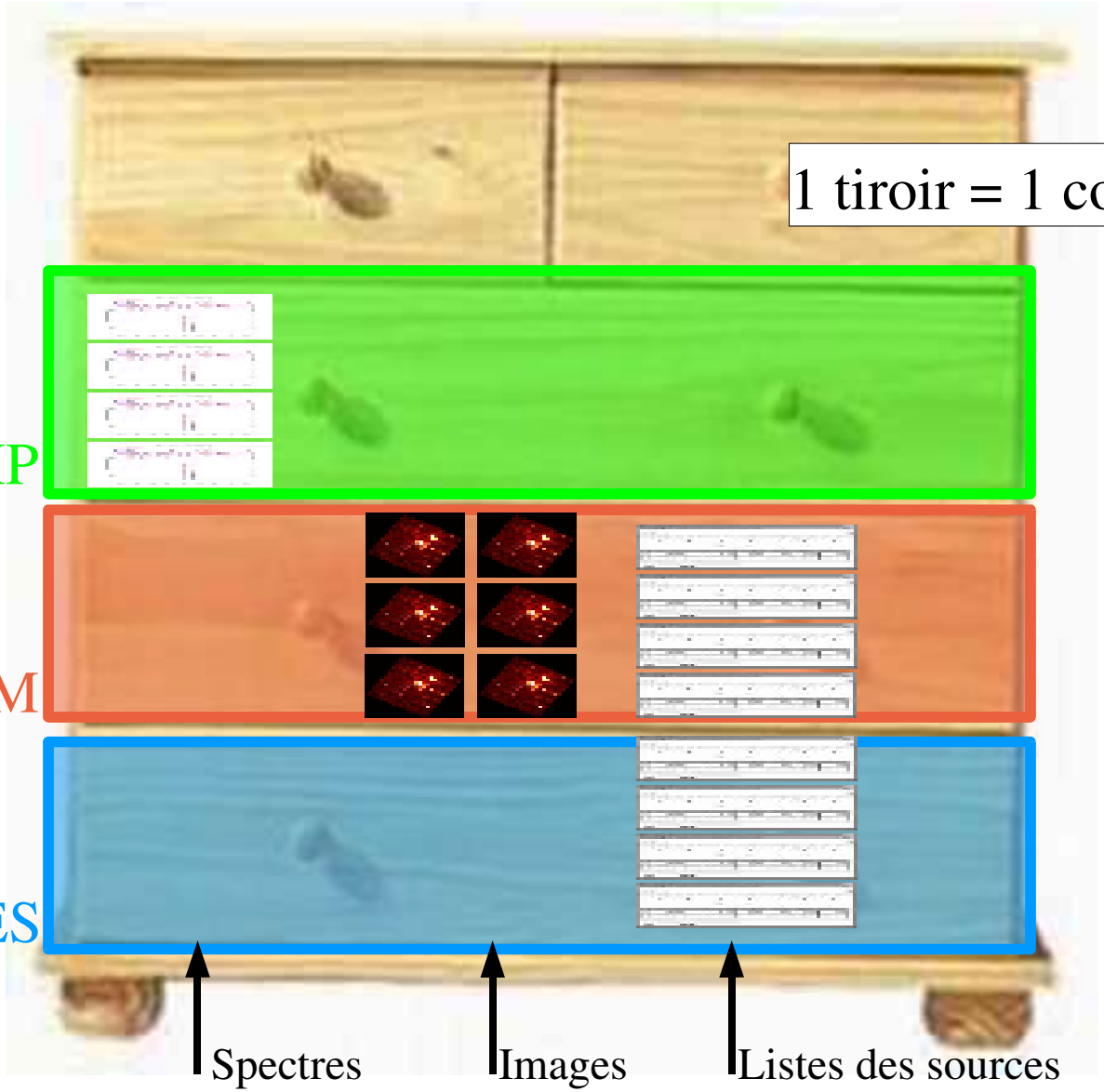
SaadaDB générique

Fichiers FITS (VOTables)

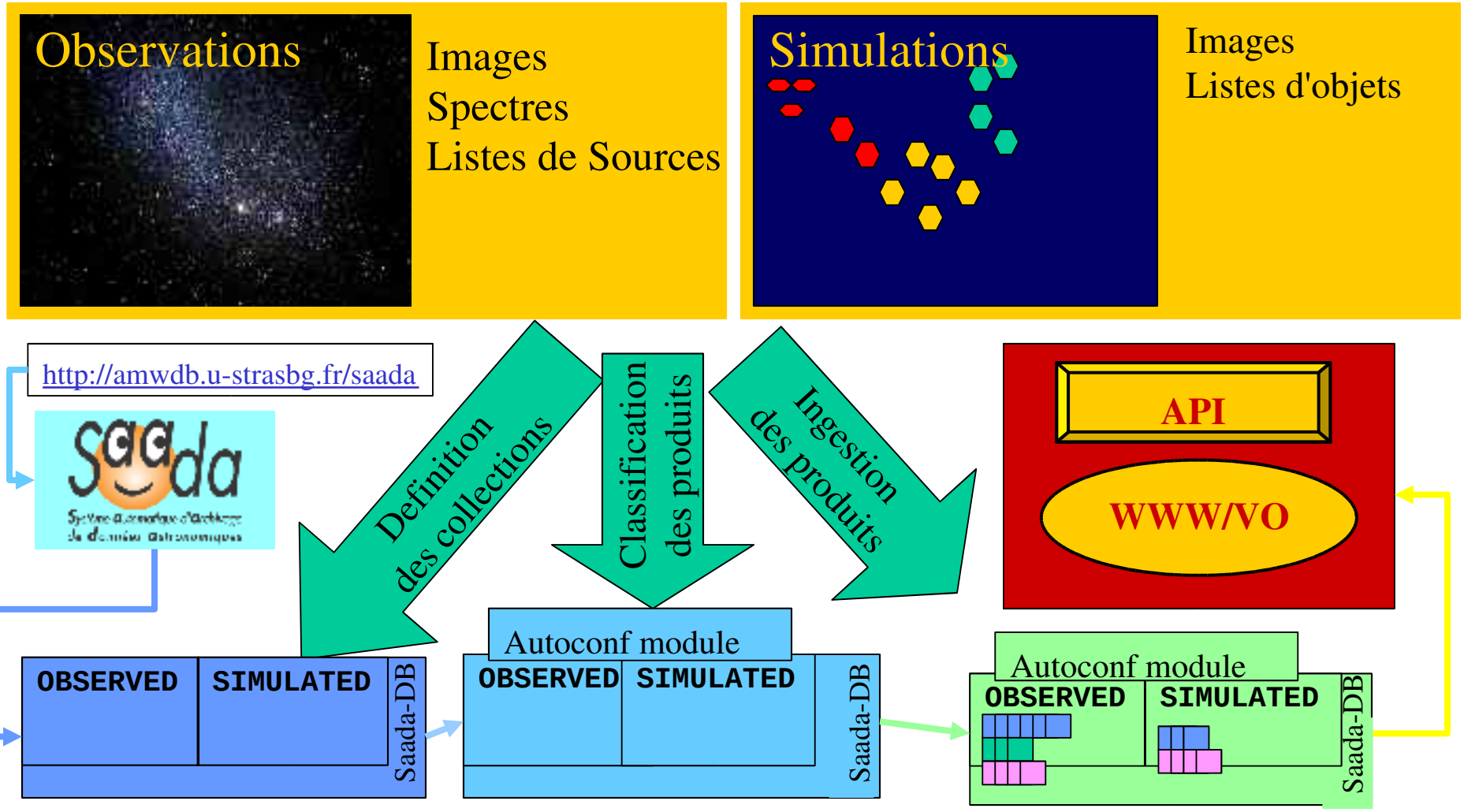
Téléchargement + installation

JDBC

Organiser la SaadaDB suivant des critères scientifiques



Etapes de création d'une SaadaDB



Organisation des données: La collection

- **La collection est une entité nommée**
- **Le contenu d'une collection est défini par l'opérateur**
  - Données d'un même instrument
  - Données d'un même modèle
  - .....
- **La collection contient**
  - Une liste de spectres
  - Une liste d'images
  - Une liste de tables avec les entrées associées
  - Une liste de « plots »
- **La collection est utilisée pour définir la portée des requêtes**
  - Requêtes sur une ou plusieurs collections
  - Tous les produits d'une même catégorie ont des attributs communs déduits des mots clés originaux

Organisation des données: La classe de produits

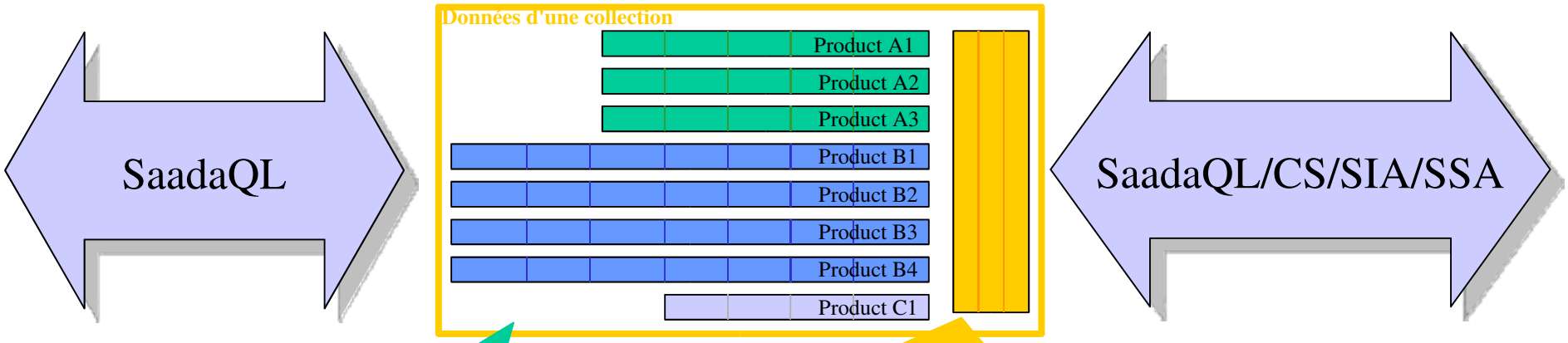
- **Les données hétérogènes dans une même collection sont groupées par classes**
- **Une classe est constituée du plus petit sur-ensemble de tous les attributs des produits qui la constituent.**
  - L'opérateur doit être vigilant quant à la fusion de produits hétérogènes dans une même classe
- **La classification des produits est semi-automatiques**
  - Règles données par l'opérateur
  - Les classes peuvent être modifiées en cours de chargement
- **On peut poser des contraintes sur les attributs d'une classe à condition que la requête ne porte que sur cette classe**

Sélectionner des données hétérogènes dans une collection

Les données présentent deux interfaces pour les requêtes

1. Une **Interface commune** pour les sélections simples
2. Les **données natives** pour les requêtes plus sélectives

Les deux interfaces peuvent être utilisées dans la même requête



Données originales
• Hétérogènes

Interface commune de données (données de collection)
• Homogènes
• Recopiées et formatées à partir des données originales

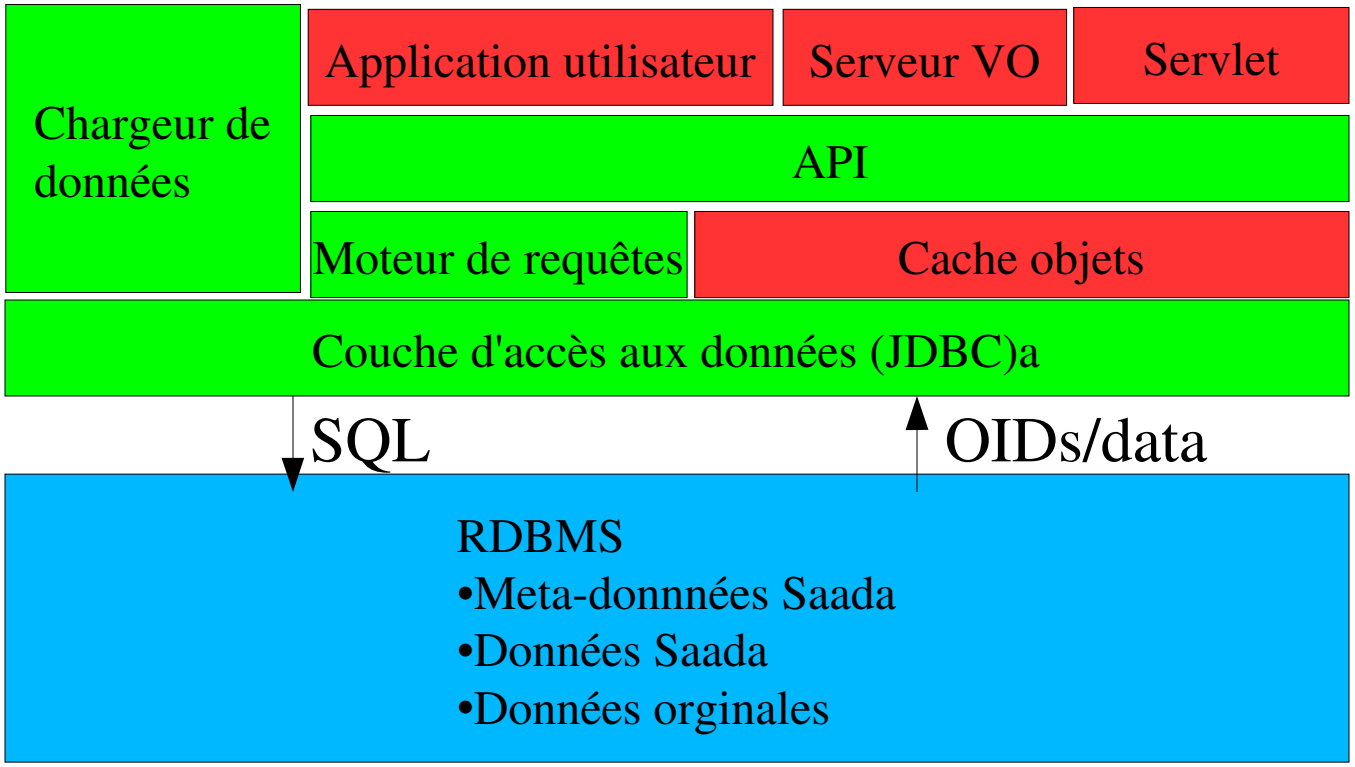
Structure d'une SaadaDB

Outils nécessaires

- Java 1.5
- J2EE 1.3 (jar)
- Axis 1.2
- Ant 1.6
- Tomcat 5
- PSQL 8

Vont être intégrés dans la distribution

Données relationnelles
Code natif
Code spécifique ou généré



SAADA

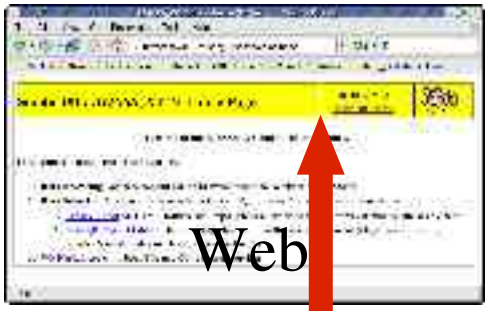
DEMO

Team: Patrick MILLAN
Laurent MICHEL
Christian MOTCH
Ngoc Hoan NGUYEN
F. Xavier PINEAU

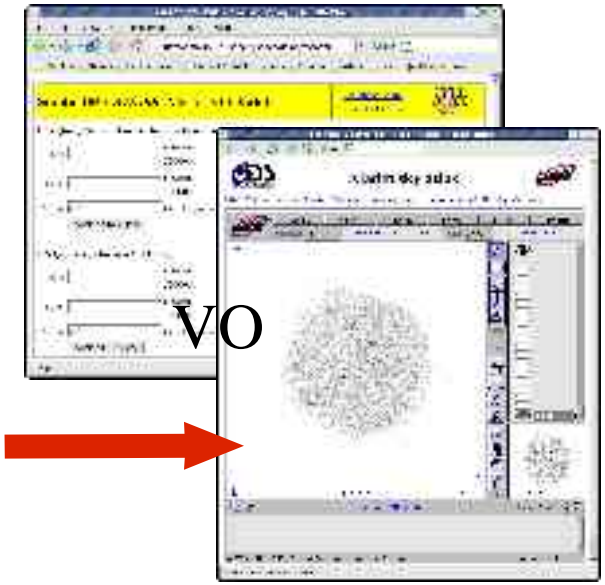
Funded by cnes Région Alsace

Supported by XMM-Newton SURVEY SCIENCE CENTRE

Principe de création d'une SaadaDB



Web



VO

API Java

SaadaDB opérationnelle

Règles de configuration

SaadaDB générique

Fichiers FITS (VOTables)

Téléchargement + installation

JDBC

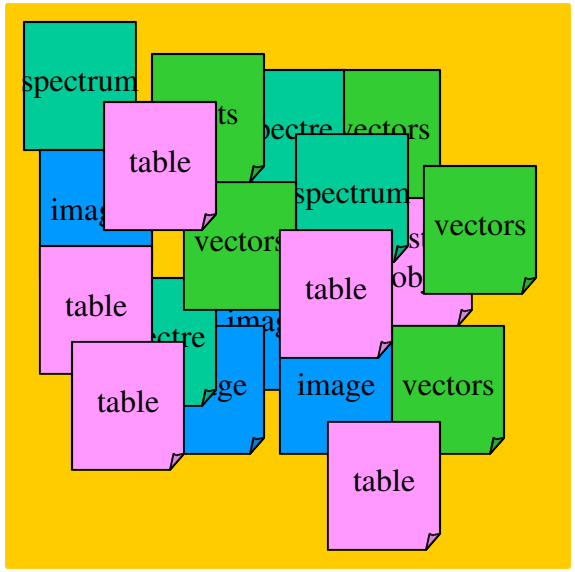
Démo 1

- **Création de la base SQL**
  - La base SQL (PSQL 8) est créée vide: `host> createdb ASOV`
- **Edition du fichier de configuration**
  - Utilisation d'un installeur (next/next/finish) 
- **Création de la SaadaDB vide**
  - commande `host> $SAADA_HOME/bin/NewSaadaDB`
  - Identification de la SaadaDB
  - Identification du système de gestion de bases de données relationnelles
  - Identification des répertoires de la base et du repository
  - Définition du système de coordonnées spatiales.
  - Définition des coordonnées spectrales 
- **Construction des servlets**
  - `host> saadadbs/ASOV/bin/build_servlet`

Organisation des données dans une SaadaDB

1 Ensemble hétérogène de produits

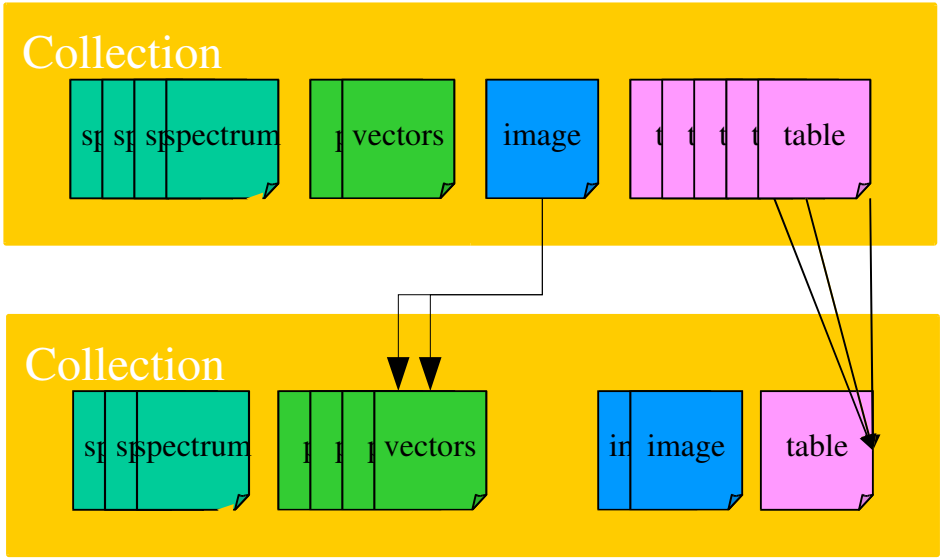
en entrée: **des fichiers FITS**



SAADA

2 Organisation en collections

En sortie: **une SaadaDB**



3 Gestion de relations permanentes

4 Requêtes sur des données hétérogènes

Organisation des données: La configuration de produits

- **Le configuration de produit est définie par l'opérateur pour charger les données**
  - Identification des fichiers relevant de la configuration
  - Choix de la méthode de classification
  - Collection d'affectation.
  - *Mapping* des attributs obligatoires (position p.e.)
- **Les utilisateurs ne voient pas les configurations.**
  - Ils ne voient que les classes et les collections.

Démo 2: Chargement des données

- **Définir les collections**
  - De nouvelles collections peuvent être ajoutées à tout moment
- **Définir les configurations**
  - Une configuration ne peut plus être modifiée depuis l'interface graphique une fois enregistrée.
  - La modification manuelle reste possible mais avec d'éventuel effets de bord.
- **Charger les produits**
  - Les produits peuvent être chargés en plusieurs fois
  - Le schéma de classe s'adapte en cours de chargement

L'interface graphique

- **L'interface graphique est basée sur des servlets Java**
  - Compilation des servlets lors de la création de la base
  - Déploiement manuel après chaque nouveau chargement
- **Accès aux données:**
  - Par produit (OID)
  - Par classe de produits
  - par collection
  - Sur requêtes
  - Par protocoles VO

Démo 3:

- **Déploiement de l'interface graphique**
  - %> buildservelt
  - %> DeployTomcat
- **Présentation des principales pages**

Le langage de requêtes SaadaQL

- **Pourquoi un langage propre**
  - Sélection de données hétérogène
  - Utilisation des collection dans les requêtes
  - Contraintes sur les motifs de corrélations
- **SaadaQL n'est pas un vrai langage...**
  - Ni le temps ni les compétences pour développer un vrai langage comme OQL
    - Grammaire complexe
    - Moteur de requêtes
- **... mais une sorte de langage *créole***
  - Mélange d'expressions propres et d'expressions SQL

Le langage de requêtes SaadaQL

On ne précise que la catégorie de produits recherchés, pas les colonnes

```
          SPECTRUM
          IMAGE2D        *                    *
Select    TABLE     From Class1          In Coll1
          ENTRY          Class1,Class2        Coll1, Coll2
          PLOT
```

La portée de la requête est définie par rapport aux collection et aux classes

```
WherePosition {
    isInCircle("M31"                          ,0.1,J2000,FK5),
    isInCircle("12:23:56 -1:2:3.5",0.2,J2000,FK5),
    isInCircle("12.23 -34.87"      ,0.2,J2000,FK5),
    }
WhereAttributeSaada {
    namesaada = 'M31'
    }
```

Supporte les recherches multi-positions

Clause WHERE SQL ne pouvant contenir que des contraintes sur les attributs de collection

Requêtes SaadaQL sur une seule classe

La requête porte sur une seule classe, donc sur une seule collection.

```
Select   SPECTRUM
         IMAGE2D
          TABLE  From Class1  In Coll1
         ENTRY
          PLOT
WhereAttributeClass {
    _Vmag > 8
    }
WherePosition {
    isInCircle("M31"                    ,0.1,J2000,FK5),
    isInCircle("12:23:56 -1:2:3.5",0.2,J2000,FK5),
    isInCircle("12.23 -34.87"       ,0.2,J2000,FK5),
    }
WhereAttributeSaada {
    namesaada = 'M31'
    }
```

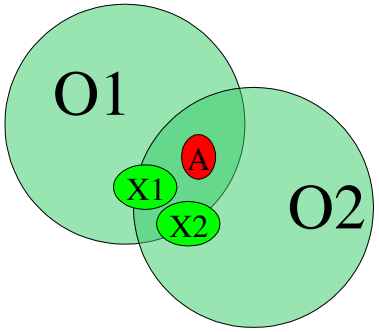
Clause WHERE SQL ne pouvant contenir que des contraintes sur les attributs de la classe

Les relations permanentes

- **Saada sait gérer des liens entre enregistrements**
  - Relations persistantes
  - Liens qualifiés
- **Intérêt des liens persistants**
  - Modélisation de la connaissance
  - Orientation fouille de données
- **Utilisation des liens**
  - Navigation entre objets associés
  - Sélection de d'objets par la nature des objets associés et la nature des liens eux-même

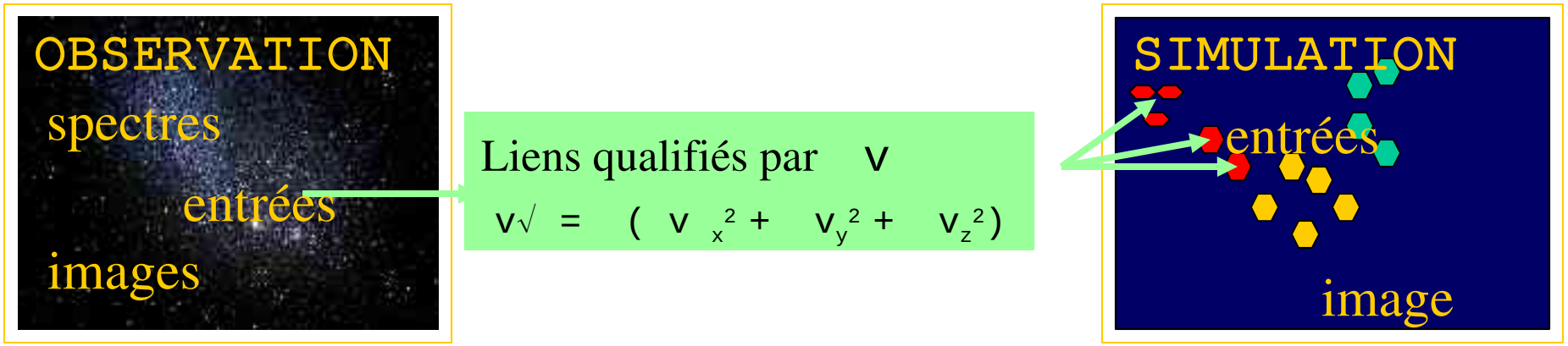
Les relations permanentes: Exemple 1

- **Les données XMM-Newton**
  - Un corpus de données à ingérer
  - Cross corrélations faites dans le pipeline
    - Corrélation entre les sources XMM et 202 catalogues
  - Liens de corrélation sont stockés dans la base
    - L'unicité des sources d'archive est restaurée.
      - X1 vue dans l'observation O1
      - X2 vue dans l'observation O2
      - A figurera dans les produit de O1 et de O2



- **Sélection de sources X par des contraintes sur les contre-parties**
  - Identification d'objets rares: outlayers et autres moutons à 5 pattes
  - Sélection de classes d'objets

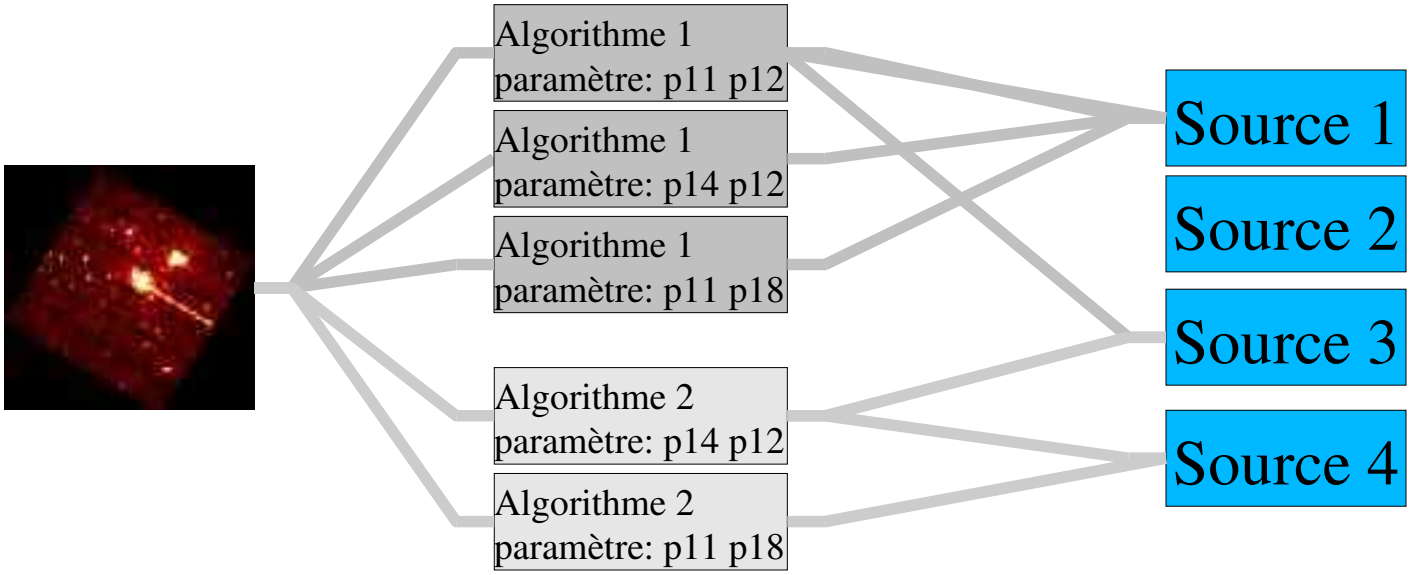
Les relations permanentes: Exemple 2



- **Liens entre données simulées et données d'expérience**
  - Association par des paramètres autres que la position (vitesse, brillance...)
  - Validation manuelle des liens
  - Annotation des liens (vraisemblance, indicateur de qualité)
- **Utilisation**
  - Sélection des données simulées validées ou non

Les relations permanentes: Exemple 3

- **Archivage de modes d'extractions**
  - Les sources sont liées aux images où elles ont été détectée
  - Les liens représentent les algorithmes de détection et leurs paramètres



Définition d'une relation permanente dans Saada

- **La relation permanente est une entité nommée dans Saada**
  - Elle relie potentiellement toutes les données d'une certaine catégorie dans une collection donnée avec tous les objets d'une certaine catégorie dans une collection donnée.
    - Exemple: toutes les images de la collection HST avec tous les spectres de la collection OHP
  - Les liens composant la relation sont définis par des qualificateurs
    - Valeur affectées au lien.
  - Saada fournit une classe Java *template* pour mettre à jour la relation. Elle doit être complétée par l'opérateur.
    - La relation peut être mise à jour par toute autre méthode.

- **Les relations permanentes ont leur propre système d'indexation afin d'éviter les jointure SQL lourdes dans les requêtes**

Démo 4: création d'une relation

- **Création de la relation permanente `SpectraToImage` liant les spectres avec les images ou ils ont été vus.**
  - Les liens sont qualifiés par la position RA/DEC du spectre
- **Des algorithmes standards seront bientôt proposés.**
  - distance
  - écart à une valeur pivot
  - comparaison d'attributs
  - ...

Utilisation des relations dans les requêtes

- **Utilisation des motifs de corrélation pour sélectionner les objets**
  - Nombre de liens (cardinalité)
  - Valeur de qualificateurs
  - Attributs des objets pointés par les liens

- **Besoin d'un opérateur spécifique dans le langage de requête**
  - Justification de SaadaQL

Expression des motifs de corrélation dans SaadaQL

```
WhereRelation {
   matchPattern {« relation »
      Cardinality(« [] », 1, 2)
      Qualifier(« RA », « [] », 12, 245)
      Qualifier(« DEC », « [] », -20 +20)
      AssObjAttSaada(« namesaada like 'K%' »)
      AssObjClass(« class1 », ...)
      AssObjAttClass(« _Vmag > 8 »)
     }
 }
```

Contrainte sur les attributs de classe de la contrepartie (si une seule classe spécifiée)

Contrainte sur la classe de la contrepartie

Contrainte sur les attributs Saada de la contrepartie

Contraintes sur les valeurs des qualificateurs

Nombre de liens devant satisfaire les conditions qui suivent

Relation sur laquelle s'applique le motif

Saada et le VO

- **Saada permet publier des données locales dans le VO**
  - Implémentation des services
    - Données de Saada visibles par les outils VO
    - Chargement dans une SaadaDB de données extraites d'autres centres de données
  - Possibilité de confronter les donner de la SaadaDB avec d'autres données
    - Crossmatch
    - Images multiplans
    - Images multibandes
- **Services implémentés**
  - Simple Image Access Protocole (SIAP)
  - Cone Search
  - Simple ADQL
  - Simple Spectra Access Protocole (SSAP)

Saada et le VO: Déploiement des services

- **L'interface graphique dispose d'une page VOPortal donnant accès aux services VO**
- **SIA et le Cone Search sont automatiquement déployés lors de la mise en service de l'interface graphique**
- **Le Web Service ADQL doit être installé manuellement**
  - Installation du gestionnaire de WS Axis dans Tomcat
  - Installation du WS ADQL
- **La déclaration de la SaadaDB dans un registry doit être faite manuellement**

Demo 6: VO

- **Utilisation de SIA**
- **Utilisation de CS**
- **Accès par Aladin**
-

Les UCDs dans SaadaQL

- **Selection dans toute la base d'entrées ayant une vitesse radiale et une magnitude V dans une certaine echelle.**

```
Select ENTRY From * In *
WhereUCD  {
        [src.veloc.hc]          > 10[km/sec]
    and [src.veloc.hc]          < 100[km/sec]
    and [phot.mag;em.opt.V]     < 20
    }
```

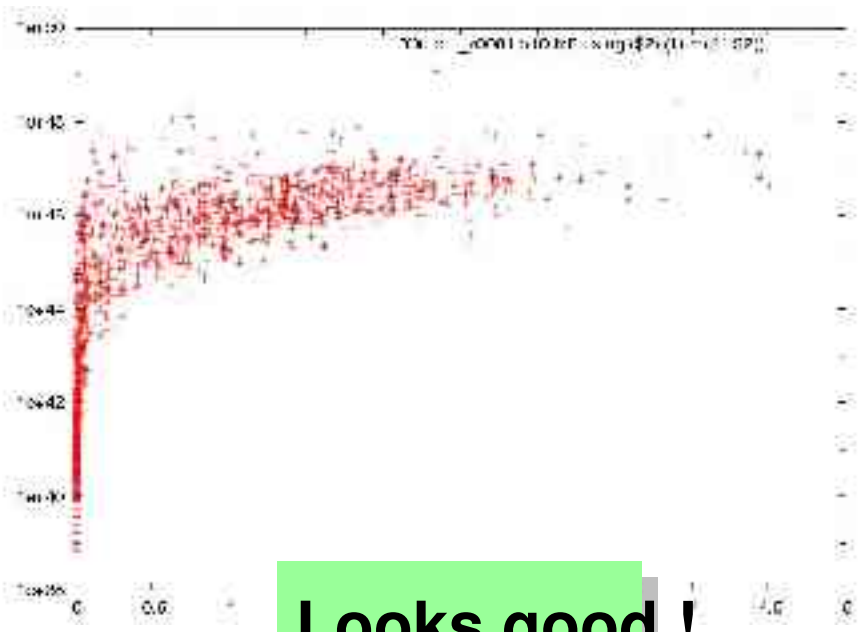
Specific SaadaQL 'where' statement

UCDs used as attributes

Unit associated with range values

Validation: The result

```
Select ENTRY From * In EPIC
WhereRelation {
  matchPattern{«EpicSrcToArchSrc»
   ,AssUCD([src.redshift] > 0.001
       and [src.redshift] < 5)
  }
}
```



Looks good !

2047 sources X sélectionnées.

Les flux X sont convertis en luminosité en appliquant un modèle d'univers plat.

Fin

Team: **Patrick MILLAN**
Laurent MICHEL
Christian MOTCH
Ngoc Hoan NGUYEN
F. Xavier PINEAU

Funded by

Supported by