Projet SoCQ: Requêtes continues orientées services pour les environnements pervasifs

Service-**o**riented **C**ontinuous **Q**uery

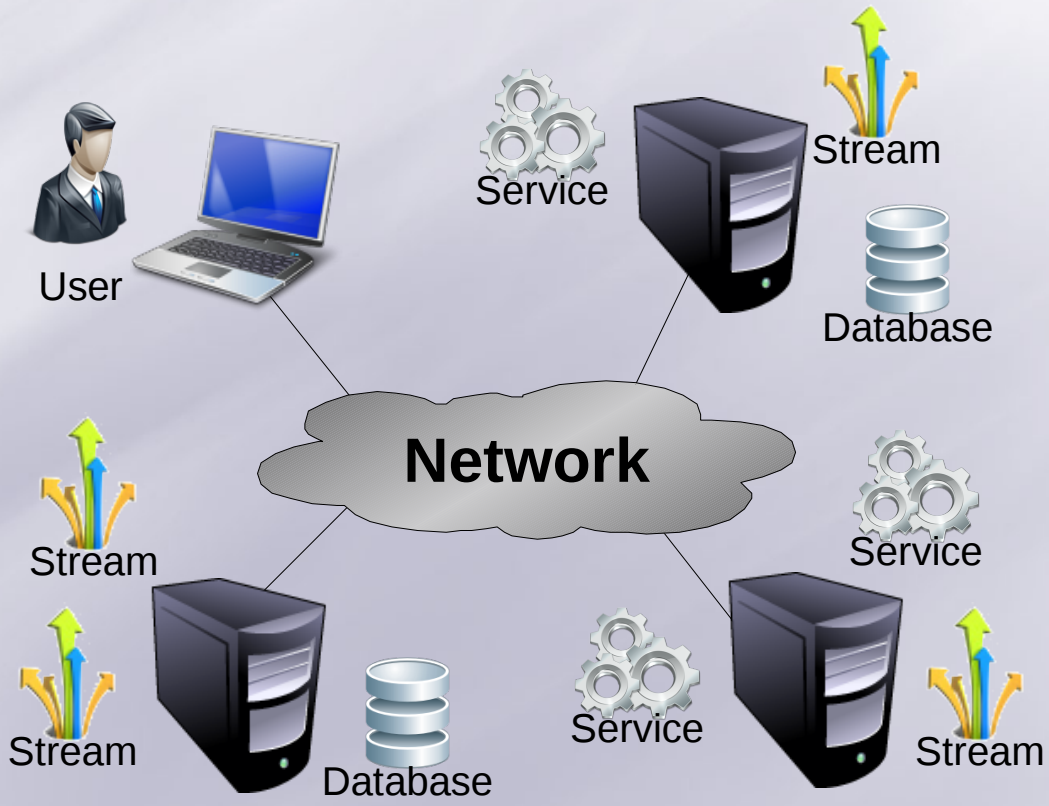
Yann Gripay, Frédérique Laforest, Jean-Marc Petit

Laboratoire d'InfoRmatique en Image et Systèmes d'information
Université de Lyon, Insa-Lyon, LIRIS – UMR 5205 CNRS
7 avenue Jean Capelle
F-69621 Villeurbanne, France
http://liris.cnrs.fr

23 janvier 2009

Contexte

- Environnement dynamique & réparti
  - Sources de données
  - Fonctionnalités
- Développement *ad hoc* d'applications "pervasives"

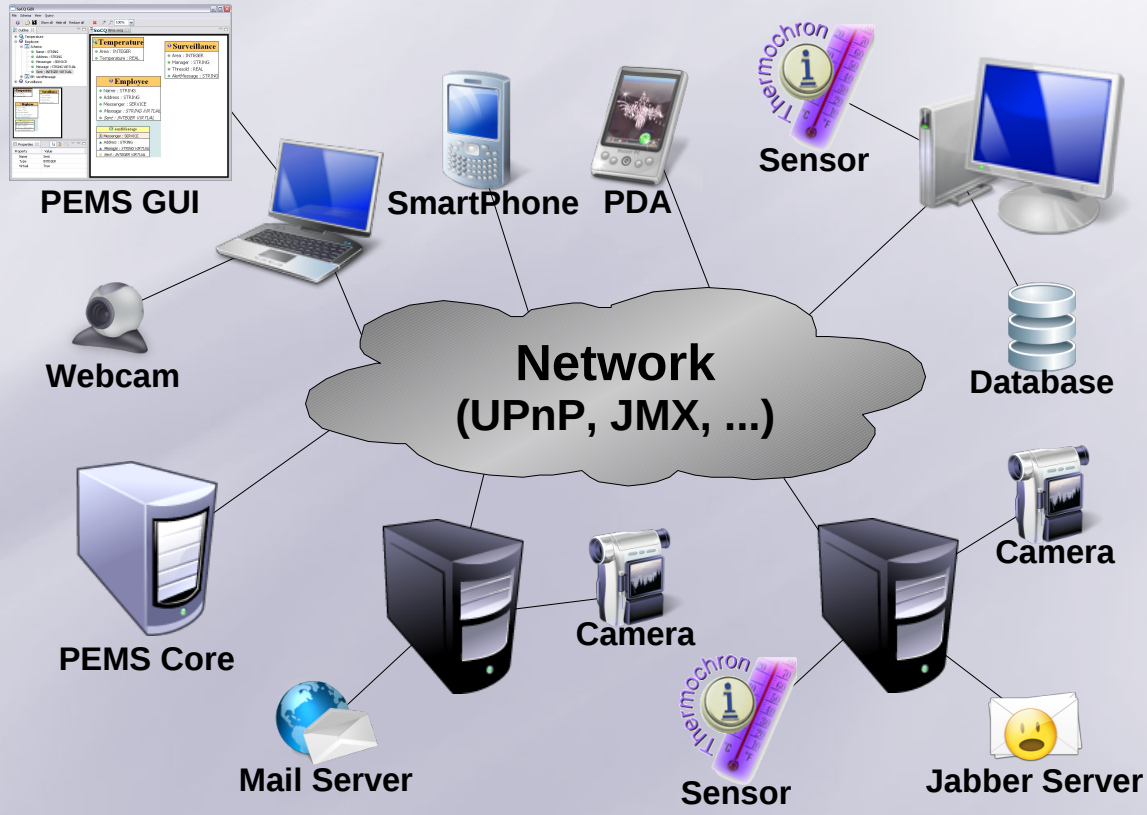


Approche & Objectifs

- Approche orientée données
  - Technologies “Bases de Données” pour le développement d’applications pervasives
- Modèle de données pour les environnements pervasifs
  - Représentation homogène des ressources *(structure)*
    - Relations
    - Flux
    - Services
  - Définition déclarative de requêtes *(langage)*
    - Algèbre
    - Langage à la SQL
- Système de Gestion d’Environnement Pervasif
  - PEMS : **P**ervasive **E**nvironment **M**anagement **S**ystem

Scénario

- Surveillance de Température
  - Données administratives
  - Capteurs de Température, Caméras
  - Messagerie



Positionnement

- Interactions données / services
  - Projet ActiveXML [S. Abiteboul, I. Manolescu *et al.*, *VLDB 2008*]
  - [W. Xue, Q. Luo, *CIDR 2005*]
- Binding Patterns
  - [D. Florescu *et al.*, *SIGMOD'99*]
  - [R. Goldman, J. Widom, *SIGMOD'00*]
- Requêtes continues
  - Projet STREAM [J. Widom *et al.*, *Université de Stanford*]
  - TelegraphCQ [S. Chandrasekaran *et al.*, *CIDR 2003*]
  - ...

Plan

Plan

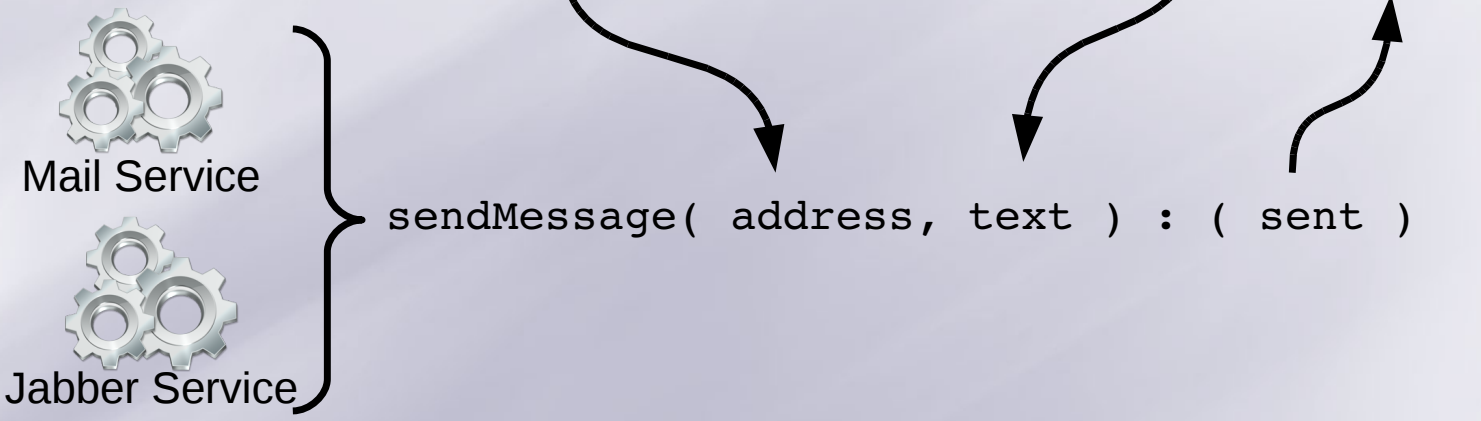
Intégration Sources de Données / Fonctionnalités

NAME	*ADDRESS*	*PROTOCOL*
Nicolas	nicolas@elysee.fr	e-mail
Carla	carla@elysee.fr	e-mail
François	francois@im.gouv.fr	xmpp


```
sendMessage( address, text ) : ( sent )
```

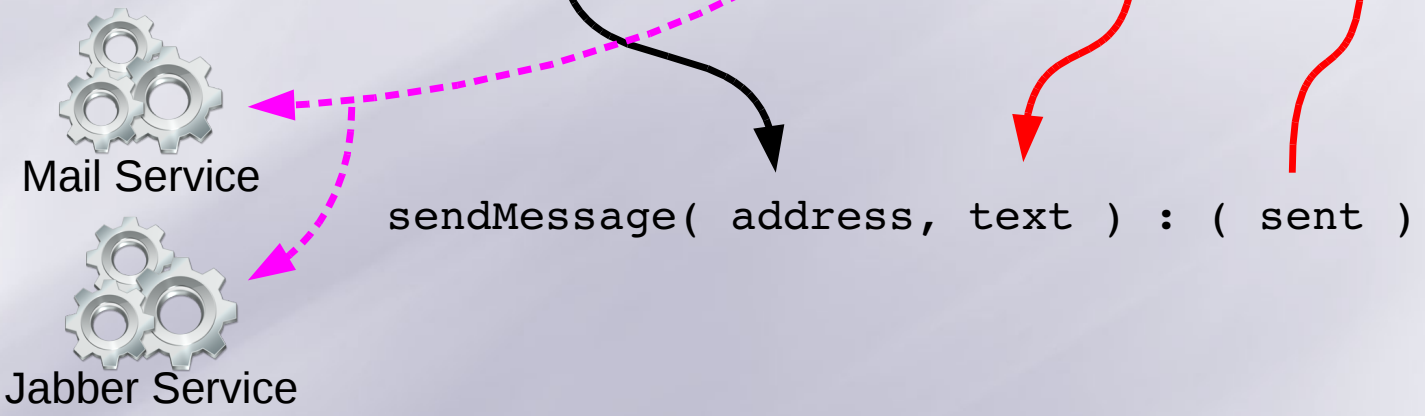
Intégration Sources de Données / Fonctionnalités

NAME	*ADDRESS*	*PROTOCOL*
Nicolas	nicolas@elysee.fr	e-mail
Carla	carla@elysee.fr	e-mail
François	francois@im.gouv.fr	xmpp



Intégration Sources de Données / Fonctionnalités

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*



Intégration Sources de Données / Fonctionnalités

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Représentation “relationnelle” des fonctionnalités

- Prototype
  - Abstraction d’une fonctionnalité
  - *Exemple:* prototype $sendMessage \in \Psi$
    - $schema(Input_{sendMessage}) = \{address, text\}$
    - $schema(Output_{sendMessage}) = \{sent\}$
- Service
  - Implémentation d’un prototype
  - Abstraction des couches systèmes
  - *Exemple:* services $mail, jabber \in \Omega$
    - $prototypes(mail) = \{sendMessage\}$
    - $prototypes(jabber) = \{sendMessage\}$

Définition des X-Relations

- Relations étendues (X-Relations)
  - Schéma étendu
    - $schema(Contacts) = \{name, address, messenger, text, sent\}$
    - $realSchema(Contacts) = \{name, address, messenger\}$
    - $virtualSchema(Contacts) = \{text, sent\}$
  - Binding Patterns
    - Hypothèse de nommage URSA !
    - $BP(Contacts) = \{\langle sendMessage, messenger\rangle\}$
- Environnement Pervasif Relationnel
  - Ensemble de X-Relations
  - *Invocation des prototypes ?*

Plan

Requêtes orientée service

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

Requêtes orientée service

NAME	ADDRESS	MESSENGER	TEXT	SENT
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	ADDRESS	MESSENGER	TEXT	SENT
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Requêtes orientée service

NAME	ADDRESS	MESSENGER	TEXT	SENT
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	ADDRESS	MESSENGER	TEXT	SENT
Nicolas	nicolas@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Requêtes orientée service

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

```
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	Bonjour!	*
François	francois@im.gouv.fr	jabber	Bonjour!	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Requêtes orientée service

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

```
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	Bonjour!	*
François	francois@im.gouv.fr	jabber	Bonjour!	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Requêtes orientée service

NAME	ADDRESS	MESSENGER	TEXT	SENT
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	ADDRESS	MESSENGER	TEXT	SENT
Nicolas	nicolas@elysee.fr	mail	Bonjour!	*
François	francois@im.gouv.fr	jabber	Bonjour!	*

sendMessage("nicolas@elysee.fr","Bonjour!")  (error)

sendMessage("francois@im.gouv.fr","Bonjour!")  (ok)

Requêtes orientée service

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	Bonjour!	error
François	francois@im.gouv.fr	jabber	Bonjour!	ok

sendMessage("nicolas@elysee.fr","Bonjour!")  (error)

mail

sendMessage("francois@im.gouv.fr","Bonjour!")  (ok)

jabber

Requêtes orientée service

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	Bonjour!	error
François	francois@im.gouv.fr	jabber	Bonjour!	ok

Requêtes orientée service

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
SELECT name,messenger,text,sent
FROM contacts
WHERE text IS « Bonjour! »
  AND name != « Carla »
INVOKING sendMessage
```

NAME	*MESSENGER*	*TEXT*	*SENT*
Nicolas	mail	Bonjour!	error
François	jabber	Bonjour!	ok

Algèbre Serena

Opérateurs

- Opérateurs relationnels
  - Sélection : $s = \sigma_F(r)$
  - Projection $s = \pi_L(r)$
  - Jointure naturelle $s = r_1 \bowtie r_2$
- Opérateurs de réalisation
  - Affectation $s = \alpha_{A \equiv B}(r)$ ou $s = \alpha_{A \equiv a}(r)$
  - Invocation $s = \beta_{bp}(r)$

Algèbre Serena

Requêtes

- Requête sur un environnement pervasif relationnel
  - Composition d'un nombre fini d'opérateurs sur un ensemble de X-Relations
- Exemple
  - `SELECT name, messenger, text, sent`
    `FROM contacts`
    `WHERE text IS "Bonjour !" AND name ≠ "carla"`
    `INVOKING sendMessage`
  - $\pi_{name,messenger,text,sent}($
    $\beta_{\langle sendMessage,messenger\rangle}(\alpha_{text\equiv\text{"Bonjour !"}}(\sigma_{name\neq\text{"carla"}}(contacts)))$
    $)$
- Optimisation
  - Problème complexe de l'équivalence
  - *Vers un modèle de coût dédié aux environnements pervasifs*

Plan

Intégration du temps

Structure – XD-Relation finie

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Intégration du temps

Structure – XD-Relation finie

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
Carla	carla@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*
Barack	barack@whitehouse.gov	mail	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Intégration du temps

Structure – XD-Relation finie

NAME	*ADDRESS*	*MESSENGER*	*TEXT*	*SENT*
Nicolas	nicolas@elysee.fr	mail	*	*
François	francois@im.gouv.fr	jabber	*	*
Barack	barack@whitehouse.gov	mail	*	*

```
sendMessage[messenger]( address, text ) : ( sent )
```

Intégration du temps

Structure – XD-Relation infinie

LOCATION	*TEMPERATURE*
Toit	9.5
Bureau	21.5
Couloir	25.0

Intégration du temps

Structure – XD-Relation infinie

LOCATION	*TEMPERATURE*
Toit	*9.5*
Bureau	*21.5*
Couloir	*25.0*
Toit	10.0
Bureau	21.5
Couloir	25.0

Intégration du temps

Structure – XD-Relation infinie

LOCATION	*TEMPERATURE*
Toit	*9.5*
Bureau	*21.5*
Couloir	*25.0*
Toit	*10.0*
Bureau	*21.5*
Couloir	*25.0*
Toit	11.0
Bureau	21.0
Couloir	24.5

Intégration du temps

Structure – Streaming Binding Pattern

SENSOR	*LOCATION*	*TEMPERATURE*
sensor17	Toit	*
sensor42	Bureau	*
sensor36	Couloir	*

```
streamTemperature[sensor]() : ( temperature ) STREAMING
```

Intégration du temps

Structure

- Relations étendues dynamiques (XD-Relations)
  - Schéma étendu
  - Défini pour chaque instant $t \in T$
  - XD-Relation finie / infinie
  - Binding pattern d'invocation / d'abonnement
- Environnement pervasif relationnel
  - Ensemble de XD-Relations

Intégration du temps

Langage

- Requêtes SoCQ sur un environnement pervasif relationnel
  - SoCQ : **S**ervice-**o**riented **C**ontinuous **Q**ueries
  - Opérateurs relationnels et de réalisation redéfinis
  - Opérateurs temporels
    - Fenêtre sur XD-Relation infinie $s = \mathcal{W}_{[size]}(r)$
    - Streaming sur XD-Relation finie $s = \mathcal{S}_{[event]}(r)$
  - Composition d'un nombre fini d'opérateurs sur un ensemble de XD-Relations
- Requêtes de découverte de services
  - Selon leur description (propriétés, méthodes, flux)

Exemple de requêtes

- Requêtes “one-shot” précédentes
  - $\beta_{\langle sendMessage, messenger\rangle}(\alpha_{text \equiv \text{"Bonjour !"}}(contacts))$
  - $\beta_{\langle sendMessage, messenger\rangle}(\alpha_{text \equiv \text{"Alerte !"}}(\sigma_{name \neq \text{"Carla"}}(contacts)))$
- Requête : fenêtre sur flux
  - ```
    SELECT *
    FROM temperatures[10]
    WHERE temperature > 35.5
    ```
  - $\sigma_{temperature > 35.5}(\mathcal{W}_{[10]}(temperatures))$
- Requête : alertes simples
  - ```
    SELECT *
    FROM temperatures [1], contacts
    WHERE text IS area
      AND temperature > 35.5
    INVOKING sendMessage
    ```
  - $\beta_{\langle sendMessage, messenger\rangle}(\alpha_{text \equiv area}(contacts \bowtie \sigma_{temperature > 35.5}(\mathcal{W}_{[1]}(temperatures))))$

Exemple de requêtes

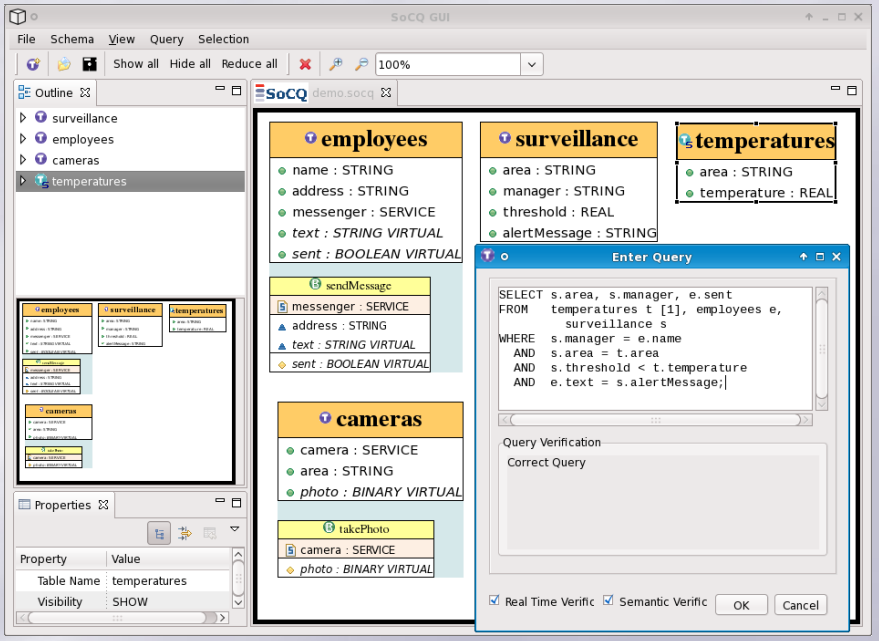
- Requête : alertes complexes
  - `Surveillance(area,name,threshold,alertMsg)`
  - ```
    SELECT *
    STREAMING UPON insertion
    FROM temperatures[1] t,
      surveillance s, contacts c
    WHERE s.area = t.area
      AND s.name = c.name
      AND c.text IS s.alertMsg
      AND t.temperature > s.threshold
    INVOKING sendMessage
    ```
  - $\beta_{\langle sendMessage, messenger\rangle}(\alpha_{text \equiv alertMsg}(contacts \bowtie \sigma_{temperature > threshold}(surveillance \bowtie \mathcal{W}_{[1]}(temperatures))))$
- Requête : découverte de services & abonnement aux flux
  - Découverte des capteurs de température
  - Abonnement aux flux de température des capteurs
    ```

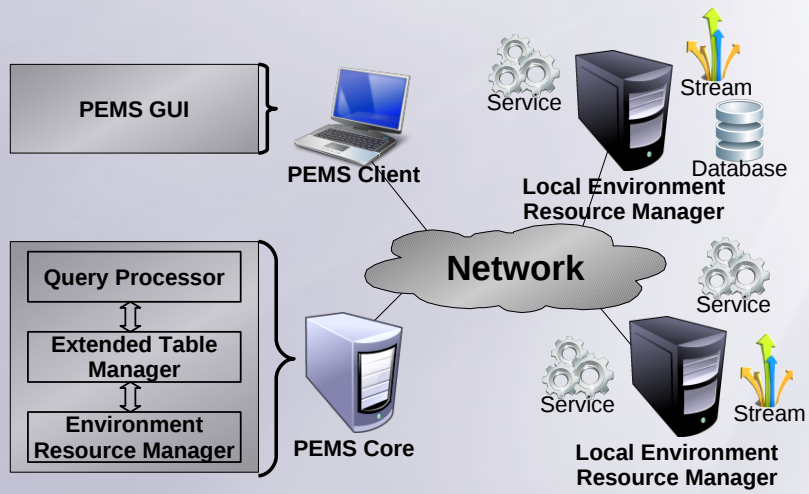
Plan

Pervasive Environment Management System

Architecture du prototype et Interface

- Local Resource Managers (Java / OSGi + UPnP)
- PEMS Core (Java / OSGi + UPnP + JMX)
- Interface Utilisateur (Java / Eclipse RCP Plugin + JMX)





Plan

Conclusion

Contributions : un environnement pervasif relationnel

- Modèle de données
  - Structure : **XD-Relations** (e**X**tended **D**ynamic **Relations**)
  - Langage : algèbre & SQL **Séréna** (**Ser**vice-**ena**bled)
  - *Optimisation des requêtes*
- Système de gestion d'environnement pervasif (PEMS)
  - Visualisation de l'environnement
  - Définition déclarative d'applications "pervasives"

Perspectives

- Projet ANR OPTIMACS
  - Service composition based framework for optimizing queries
- Vers une plus grande dynamicité
  - Adaptation dynamique des requêtes
  - PEMS pair-à-pair

MERCI DE VOTRE ATTENTION !

SoCQ Project Web Site

http://socq.liris.cnrs.fr