DESIGN OF A DECLARATIVE LANGUAGE FOR PERVASIVE SYSTEMS

20 dicembre 2007

Tesi di Laurea Specialistica:

Fortunato Marco

Marelli Marco

Relatore:

Prof. Schreiber Fabio A.

Co-relatore:

Ing. Camplani Romolo

Introduction

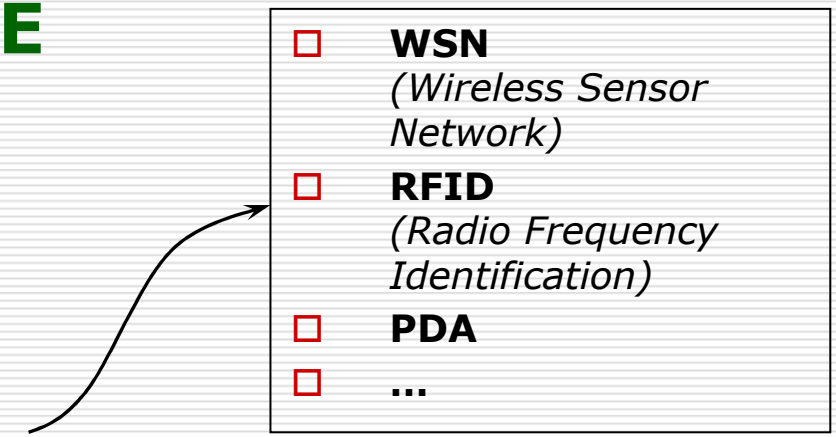
DESIGN

of a

DECLARATIVE LANGUAGE

for

PERVASIVE SYSTEMS



Existing similar projects (1)

Supported devices

- Type of nodes (*WSN, RFID, PDA, **PERVASIVE SYSTEM***)
- Supported heterogeneity level

		Deploy-time heterogeneity	
		Little	**Full**
Run-time heterogeneity	**Little**	Homogeneous systems	Partially Homogeneous systems
	Full	-	***Heterogeneous systems***

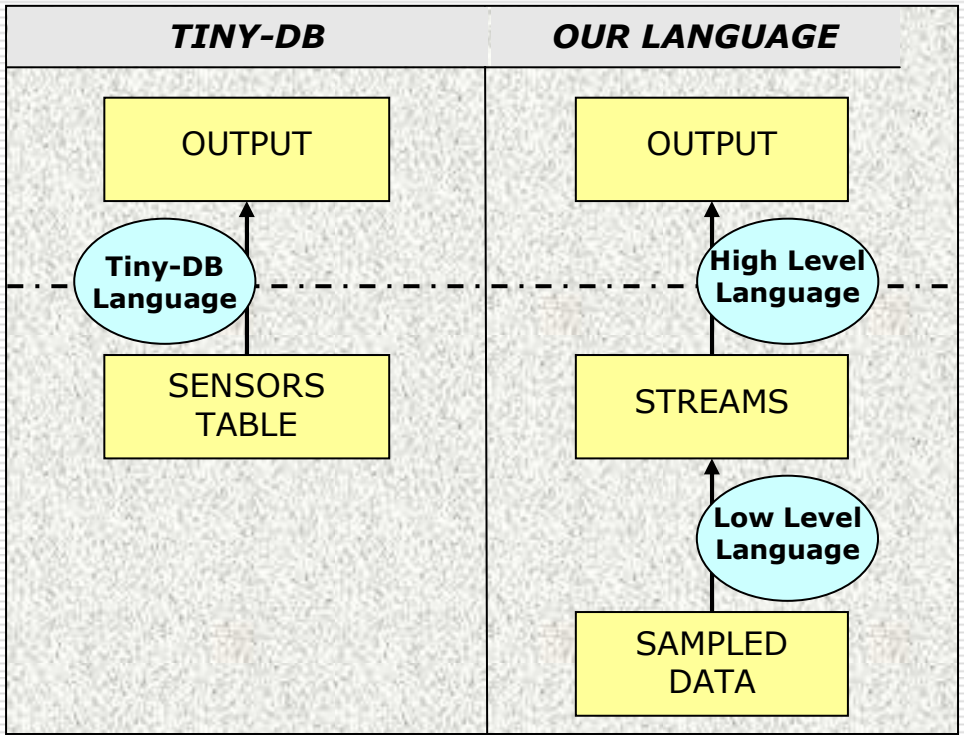
Programming model

- Declarative languages *(Database abstraction of the network)*
  - ***SQL LIKE*** (TinyDB, Cougar, …)
  - Non SQL Like (DSN, …)
- Procedural languages
  - Virtual Machine (Maté, Magnet)
  - Macroprogramming (Kairos)

Existing similar projects (2)

- ***Important projects***

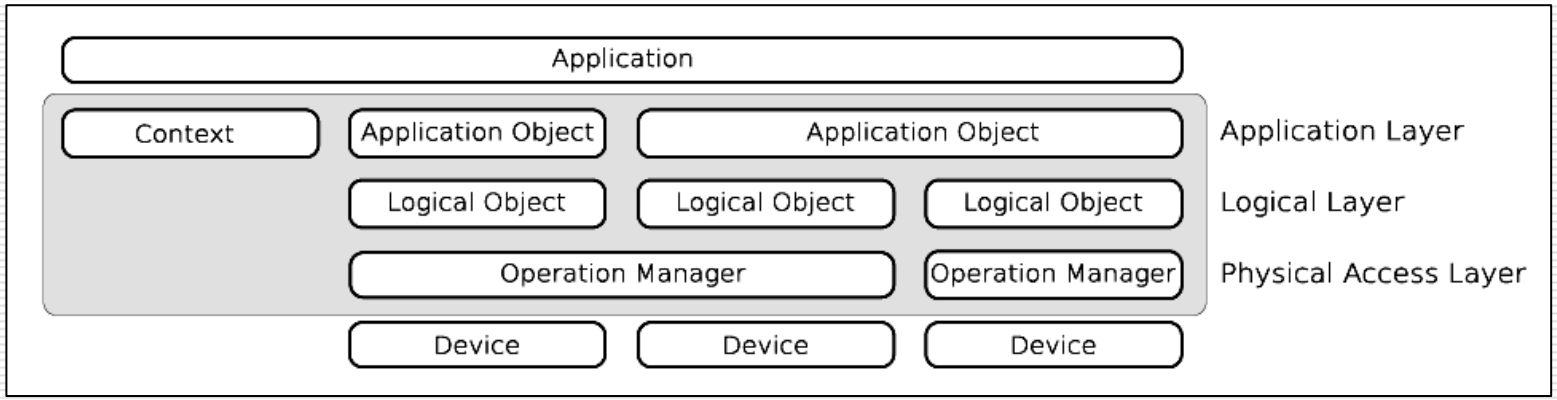
  - Tiny DB
    - Introduced the idea of abstracting a WSN as a database
  - GSN
    - It is the most similar project to ours (both in the goals and in the approach)



Features of our approach

- ***Network abstraction***
  - Provide a database view of the network
  - Hide low level characteristics of physical devices
- ***Functional features***
  - Set sampling parameters (e.g. sample rate)
  - Manipulate sampled data to produce query results
- ***Non functional features***
  - Many non functional constraints can be identified:
    - to decide if a node should participate to a query
    - to set the current sample rate
    - to set the rate used for sending data out of the node
    - to retrieve information about network nodes
  - A generalized management is required

Architecture for pervasive systems



- *Application Layer*: Front end for data access
- *Logical Object Layer*: Abstraction for physical devices
- *Device Access Layer*: Sw infrastructure for device access

Logical objects (1)

- Logical object functionalities
  - ***Retrieve attributes***
    - ***STATIC*** attributes
      - *ID, device_type, maximum_sampling_rate, location (fixed devices)*
    - ***PROBING DYNAMIC*** attributes
      - *temperature, pressure, location (mobile devices), power_level*
    - ***NON PROBING DYNAMIC*** attributes
      - *last_sensed_RFID_reader*
  - ***Fire notification events***
  - ***Get the list of supported attributes***

Logical objects (2)

- Abstraction of the sampling operation:
  - ***PERIODIC SAMPLING***
    - Reading of a logical object attribute periodically
  - ***EVENT BASED SAMPLING***
    - Reading of a logical object attribute after an event is raised

Logical objects (2)

- Example: for RFIDs, two abstractions are possible:
  - ***RFID TAG as a sensor***
    - Sampled data: Id of the last reader which sensed the tag
  - ***Reader as a sensor***
    - Sampled data: Id of the last tag sensed by the reader
  - In both cases the sampling is "***event based***": when a tag is sensed by reader, the logical object wrapping the tag (or the reader) raises an event "last_reader_changed" (or "last_tag_changed")

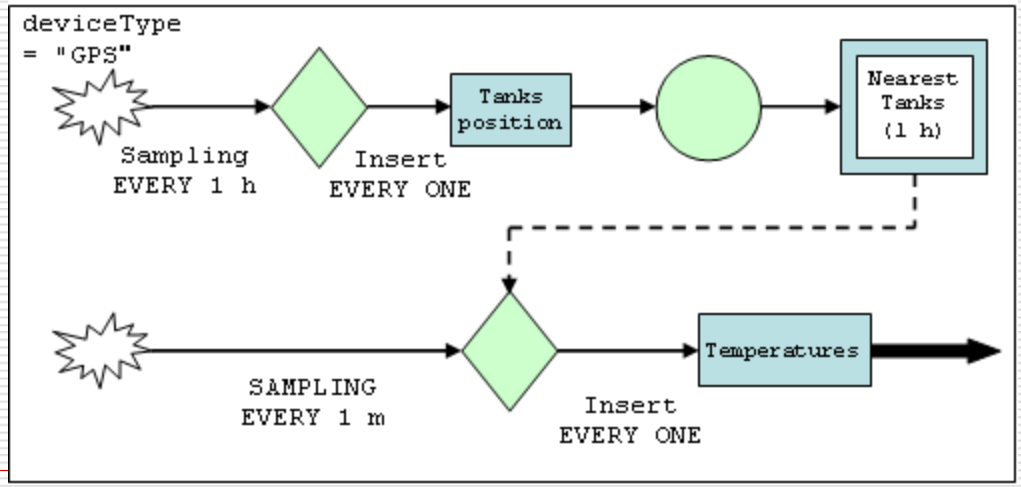
Language design

```
CREATE STREAM TanksPositions (gpsID ID, linkedBaseStationID ID, distanceFromP FLOAT) AS
LOW:
   EVERY ONE
   SELECT ID, linkedBaseStationID, dist_from_P(locationX, locationY)
   SAMPLING EVERY 1 h
   EXECUTE IF deviceType = "GPS"


CREATE SNAPSHOT NearestTank (gpsID ID, linkedBaseStationID ID)
WITH DURATION 1 h AS
HIGH:
   SELECT TanksPositions.gpsID, TanksPositions.linkedBaseStationID
   FROM TanksPositions (1 h)
   WHERE TanksPositions.distanceFromP = MIN(TanksPositions.distanceFromP)


CREATE OUTPUT STREAM Temperatures (sensorID ID, temp FLOAT) AS
LOW:
   EVERY ONE
   SELECT ID, temp
   SAMPLING EVERY 1 m
   PILOT JOIN NearestTank ON NearestTank.linkedBaseStationID = baseStationID
```

- User submitted query
  - Data structures
    - *Streams*
    - *Snapshots*
  - *Queries*
    - *Low level queries*
    - *High level queries*



Low level queries (1)

Sample the temperature every 30 seconds and, every 10 minutes, report the number of samples that exceeded a given threshold

INSERT INTO STREAM Table **(**sensorID**,** temperature**)**
LOW:

EVERY 10 **m**
SELECT ID, COUNT(temp**,** 10 **m)**

SAMPLING
EVERY 30 **s**
WHERE temp **> 100**

EXECUTE IF
powerLevel **>** 0.2 **AND EXISTS (**temp**)**

DATA MANAGEMENT SECTION

Event based activation

Time based activation

SAMPLING SECTION

Event based sampling

Time based sampling

EXECUTION CONDITIONS SECTION

Low level queries (2)

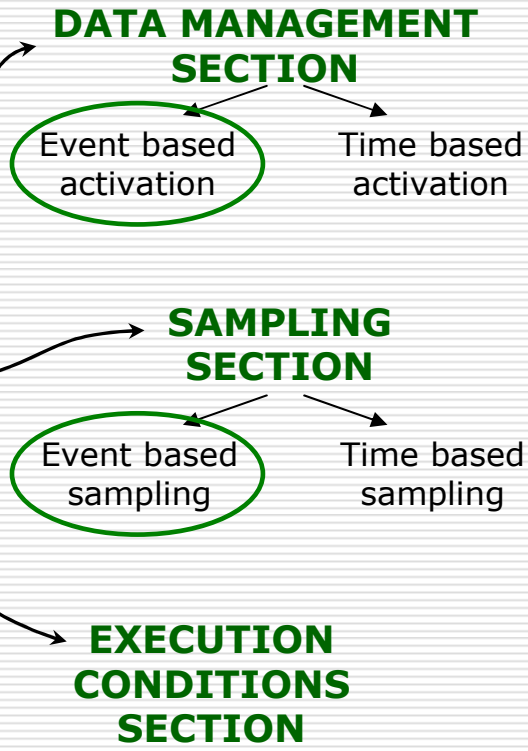
Produce e record whenever the tag with ID [tag] is sensed by a reader in the system

INSERT INTO STREAM Table **(**readerID**)**
LOW:

EVERY ONE
SELECT lastReaderID

SAMPLING
ON EVENT lastReaderChanged

EXECUTE IF ID = *[tag]*



Pilot join (1)

- **The *PILOT JOIN* operation activates the execution of a low level query on logical objects conditioned by values sampled on *OTHER NODES***

- Example:
  - Monitor the temperature of all the pallets in trucks whose current position is in a given parking area
    - Temperature sensors on pallets
    - Position sensors on trucks

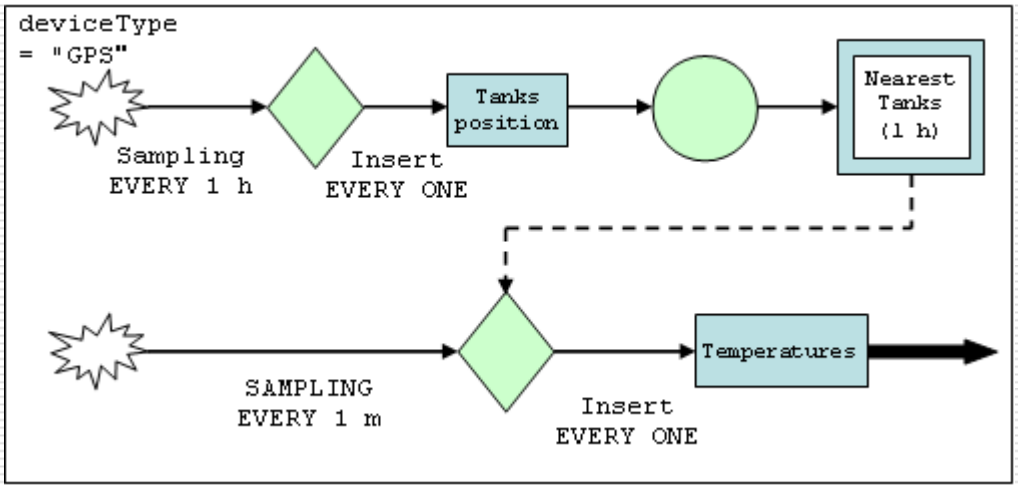
```
PILOT JOIN BaseStationList ON
        currentBaseStation = baseStationList.baseStationID
```

Pilot join (2)

- Two types of pilot join are supported:
  - ***EVENT BASED*** pilot join
    - ***When an event happens, a given set of nodes are fired to sample*** *(*e.g.: sense pallet temperature for 15 minutes every time a truck enters parking area *B)*
  - ***CONDITION BASED*** pilot join
    - ***Continuous sampling is performed on nodes connected to a given base station*** (e.g.: start sampling every 15 minutes the temperature of pallets whose last sensed position was in parking area *B)*

Example of a complex query (1)

- *There is a set of tanks, containing some temperature sensors. A GPS and a base station are mounted on each tank.*
- *The temperature sensors contained in the tank nearest to a given point P must be sampled every minute.*
- *The distances of the tanks from the point P must be revaluated every hour.*

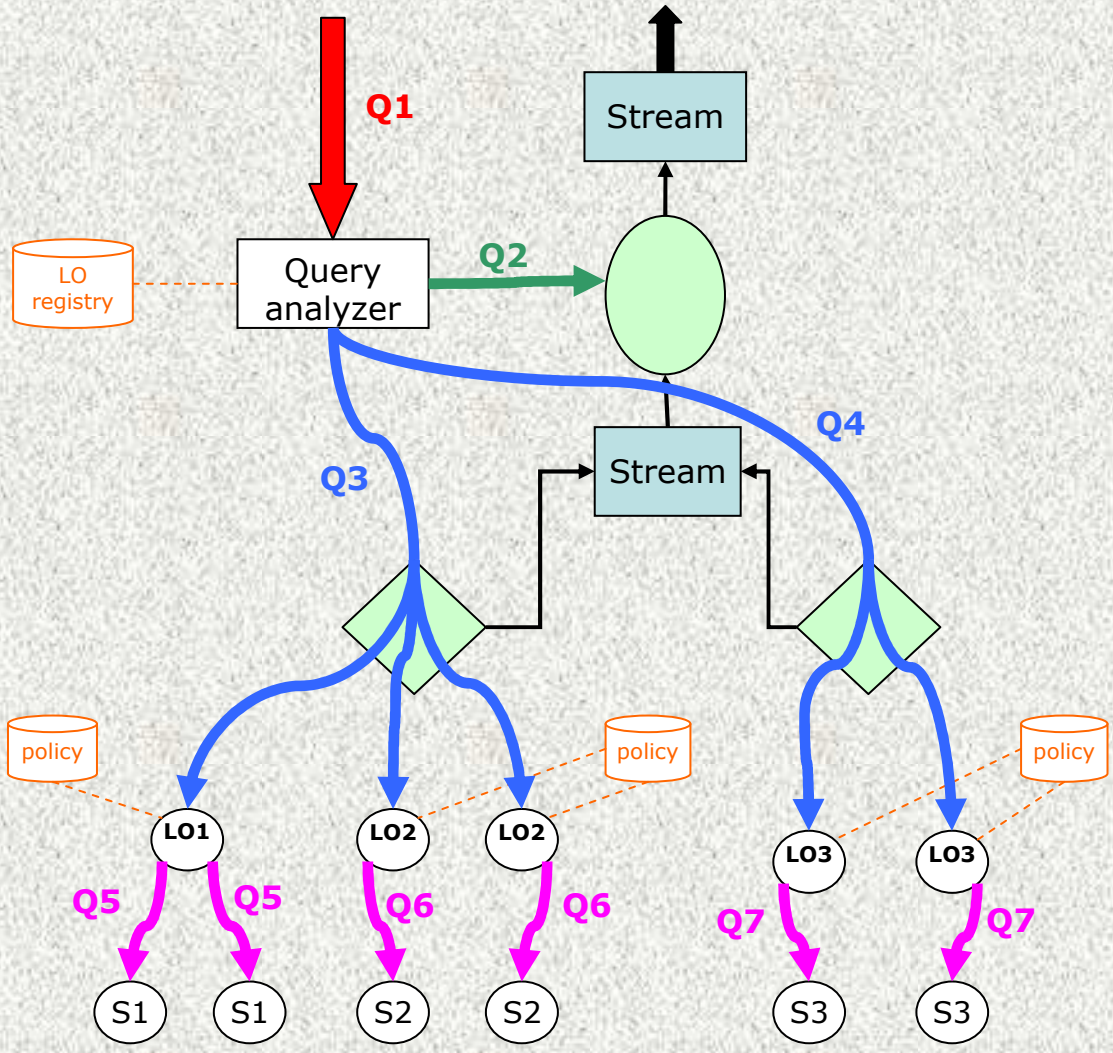


Example of a complex query (2)

GPS			
Logical object wrapping a GPS device			
Field Name	**Data Type**	**Field type**	**Description**
ID	ID	ID	Logical object identifier
linkedBaseStationID	ID	S	ID of the base station mounted over the same tank
locationX	FLOAT	P	Sensor location – X coordinate
locationY	FLOAT	P	Sensor location – Y coordinate
deviceType	STRING	S	Type of device

WSN node			
Logical object wrapping a single WSN node equipped with a temperature sensor			
Field Name	**Data Type**	**Field type**	**Description**
ID	ID	ID	Logical object identifier
baseStationID	ID	NP	ID of the base station the WSN node is currently connected to
temp	FLOAT	P	Sampled temperature

Query execution



State of art

- ***Identification of language requirements***
- ***Definition of EBNF grammar***
- ***Definition of semantics for all the language clauses***
- ***Implementation of the parser***
- ***Design of logical objects interfaces***
- ***Design and implementation of the LO registry***
- ***Implementation of execution engines***
- ***Implementation of a devices simulator***