UNIVERSITÉ LIBRE DE BRUXELLES January 19, 2016
Faculté des Sciences Appliquées

INFO-H-415 – Advanced databases

First session examination

The exam is divided in four main sections. All sub-questions are worth approximately the same amount of points. However, some of these will only take you a minute, some require a bit more thinking, and a couple would require weeks to answer perfectly. Make the best use of your time.

Schema

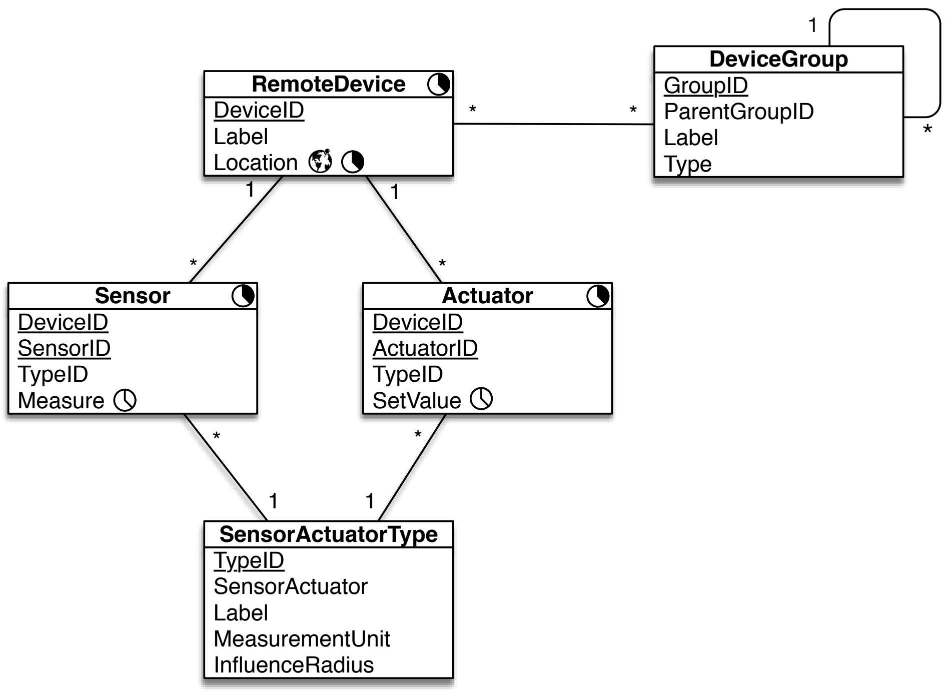
We consider a simplified domotics system for home automation and the database that allows managing it.

Physically, the system consists in a central processing unit to which a set of remote devices is connected. These remote devices are located in different places of the home and interact with their immediate environment by means of *sensors* (allowing to gather data about that environment) and *actuators* (that allow them to carry out some action in that same environment). Each remote device can be equipped with zero, one or more sensors, as well as with zero, one or more actuators.

In our setup, we do not consider "intelligent" inter-connected remote devices, but "brain-less" devices that are only connected to the central unit and react according to the latter's commands.

Example. A *thermostatic valve* remote device installed in a bed room would have at least a temperature sensor to sense the room's temperature and send that data to the central unit upon request, as well as an actuator that would set the radiator's valve according the the central unit's command. One could, however, also equip this remote device with a presence sensor that could sense whether or not a person is present in the room. ☐

The database that we consider is intended for the central unit to manage the set of remote devices that are attached to it.



Legend: 🕓 ≡ date/time, ◑ ≡ time interval, 🌍 ≡ geographic point

Each remote device is recorded in the **RemoteDevice** table and identified by its DeviceID. During the lifecycle of the system, some devices can be added or removed to/from it. Each device has a given location (e.g. the bedroom), but this location can change over time.[1] For each sensor or actuator attached to a remote device, the central unit also records all read measures or set values with the corresponding timestamp, using respectively **Sensor**.Measure or **Actuator**.SetValue.

The table **SensorActuatorType** describes a given sensor or actuator type (the attribute SensorActuator can take either value "`Sensor`" or "`Actuator`"). The table also defines the measurement unit (MeasurementUnit, e.g., "`Celsius`" for temperature) and the influence radius (InfluenceRadius). The latter attribute represents how far a sensor can sense its environment (e.g. for a presence sensor, it would represent the maximum distance from the sensor's location at which a person may be detected).

The table **DeviceGroup** allows creating a logical hierarchy of groups of remote devices. A group has a Type that indicates the type of remote devices that it contains (e.g, a value "`TempSensors`" indicates a group that contains devices that have the ability to sense the temperature). A group may (but must not) belong to a parent group identified by ParentGroupID.

0 Preliminary question (1 pt)

0.1 Define a relational schema corresponding to the conceptual schema given above. The relational schema should integrate attributes for both temporal and spatial questions below. If required, use the conventions of PostgreSQL for naming the attributes' types.

1 Active Databases (4 pt)

Note. For this question, assume that a Microsoft SQL Server database is used.

1.1 Ensure that each device in **RemoteDevice** has at most one *sensor* of each type (e.g., there may not be two temperature sensors attached to one remote device).

1.2 Ensure that a device group of type "`TempSensors`" only contains remote devices that have a temperature sensor (i.e., the corresponding TypeID entry in **SensorActuatorType** has as Type "`Sensor`" and as Label "`Temperature`"). Or stated differently: ensure that there is no remote device belonging to a "`TempSensors`" device group that is not equipped with a temperature sensor. Note that these temperature sensing remote devices may have other sensors and/or actuators.

1.3 Ensure that there is no cycle in the hierarchy of device groups (**DeviceGroup**).

1.4 The central unit is continuously recording the sensed measures (in **Sensor**). To limit the size of the database, ensure that there are at most 4000 entries (which corresponds roughly to six months of hourly recordings) for each sensor, always keeping the most recent records.

2 Temporal Databases (5 pt)

Note. For this question, assume that a Microsoft SQL Server database is used.

2.1 Ensure that there is no sensor measure nor actuation value that has been respectively read or set outside the corresponding remote device's lifecycle.

[1] We will *not* consider mobile devices that frequently change their location. Practically, this implies that the temporality of the location will be an *interval* rather than a date-time.

2.2 Ensure that at any point in time, there is always at least one sensor *or* actuator attached to a remote device.

2.3 Give the interval(s) of time when the system had the highest number of attached remote devices.

2.4 For each *sensor*, give the largest timespan (in hours) between two measures.

2.5 For each remote device, give the (coalesced) history of number of sensors and actuators. (Each row should have the following columns: DeviceID, FromDate, ToDate, SensorCount, and ActuatorCount.)

3 Spatial Databases (5 pt)

Note. For this question, assume that we are using a PostgreSQL database with the PostGIS extension loaded. In addition, unless explicitly specified, you do not need to take the data's temporality into account.

3.1 Give the label of device groups that have five or more remote devices that are located at at most 5 meters from each other.

3.2 For each remote device that is *not* equipped with a temperature sensor (cf. question 1.2) but is equipped with a radiator valve actuator (i.e., the corresponding TypeID entry in **SensorActuatorType** has as Type "`Actuator`" and as Label "`RadiatorValve`"), provide the DeviceID of the closest remote device that is equipped with a temperature sensor.

3.3 For each lowest level group of temperature sensing devices (a lowest level group is one that is not a parent of any other group; a temperature sensing group of devices has its type set to "`TempSensors`"), give the geometric shape that represents the total area covered by all temperature sensors of the group's devices. The area covered by one sensor is assumed to be a circular area of radius InfluenceRadius around the Location of the remote device it is attached to.

3.4 For each lowest level group of remote devices, give the DeviceID of the device that is located the furthest away from the group's central location. (The central location of a group is the centroid of its devices's locations.)

3.5 For each lowest level device group, give the DeviceID of the temperature sensing remote devices whose influence domain (determined by InfluenceRadius of the temperature sensor around the device's Location) has no intersection with any other remote device's "temperature sensing" influence domain of the same device group.

Command references

- `float` **`ST_Area`**`(geometry)`
  returns the area of the surface of geometry.
- `geometry` **`ST_Buffer`**`(geometry, radius)`
  returns a geometry covering all points within a given distance from the input geometry.
- `geometry` **`ST_Centroid`**`(geometry)`
  returns the geometric center of a geometry.
- `float` **`ST_Distance`**`(geometry, geometry)`
  returns the 2D Cartesian distance between two geometries.

- `boolean` **`ST_Intersects`**`(geometry,geometry)`
  returns TRUE if the Geometries/Geography "spatially intersect in 2D" - (share any portion of space) and FALSE if they don't (they are Disjoint).
- `geometry` **`ST_Union`**`(geometry_set)`
  returns a geometry that represents the union of the Geometries.

4 Object Databases (5 pt)

Note. For this question, assume that an Oracle object relational database is used.

4.1 Extend the schema to make **Sensor** and **Actuator** both derive from a **SensorActuator** class.

4.2 Write type and table definitions for the above schema in such a way that all relationships can be traversed in both directions by queries.

Now, write the queries to answer the following questions.

4.3 Give the DeviceID of all remote devices having a sensor that measures temperature in degrees Celsius, i.e., for which MeasurementUnit = “`Celsius`”.

4.4 Give the GroupID of the device groups that contain exactly one device that measures temperature in degrees Celsius, and exactly one (possibly the same) device that can actuate the temperature via a radiator valve (see question 3.2).

4.5 For each lowest-level group of devices, give the DeviceID of the device that is equipped with the sensor that has the widest influence radius in comparison to the other devices of its group.