
# DATABASES OF USER INFORMATION

Data models for the SmartH2O platform

**SmartH2O**

Project FP7-ICT-619172

Deliverable D3.1 WP3

Deliverable
Version 3.1 – July 31 2015
Document. ref.:
D31.POLIMI.WP3.V3.1

| | |
|---|---|
| **Programme Name:** | ICT |
| **Project Number:** | 619172 |
| **Project Title**: | SmartH2O |
| **Partners**: | Coordinator: SUPSI<br>Contractors: POLMI, UoM, SETMOB, EIPCM, TWUL, SES, MOONSUB |
| **Document Number**: | smarth2o. D31.POLIMI.WP3.V2.0 |
| **Work-Package**: | WP3 |
| **Deliverable Type:** | Document |
| **Contractual Date of Delivery:** | 30 September 2014 |
| **Actual Date of Revision**: | 31 July 2015 |
| **Title of Document**: | Databases of user information |
| **Author(s):** | Elisa Quintarelli, Dario Piga, Andrea Cominola, Matteo Giuliani, Andrea Castelletti, Andrea Emilio Rizzoli, Alessandro Facchini, Piero Fraternali, Chiara Pasini, Giorgia Baroffio, Ricardo Wissman-Alves, Mark Holt, Marco Bertocchi, Luigi Caldararu, Sever Calit. |
| **Approval of this report** | Approved by Project Coordinator |
| **Summary of this report:** | Literature review on past residential water end use studies that have been conducted in the last years. Based on the analysis of past water end use studies at the household level, a tentative set of the main determinants influencing water consumption have been identified and a set of variables which should be included in the SmartH2O database has been determined. This set of variable will be gradually enriched based on the interactions with water utilities and users. The structure of the SmartH2O database has been defined in terms of Entity-Relationship models, and different services allowing the interaction between users, smart meter infrastructure and SmartH2O database are proposed. |
| **History**: | |
| **Keyword List**: | Databases, Platform Model, User information. |
| **Availability** | This report is confidential |



This work is partially funded by the EU under grant ICT-FP7-619172

# Document History

| Version | Date | Reason | Revised by |
|---|---|---|---|
| 1.1 | 30/4/2015 | Initial document, based on v1.0, to start with the revision to incorporate the comments of the expert reviewers after the first review meeting | A.E. Rizzoli |
| 2.0 | 4/5/2015 | Alignment of the database conceptual model to the last revisions; insertion of the SQL code for generating the relational schema. | Piero Fraternali, Chiara Pasini, Giorgia Baroffio |
| 3.0 | 27/7/2015 | A new section 4.4 has been added. It contains the description of the data assimilation procedure and the SMDMC component. | Sever Calit, Luigi Caldararu, A.E. Rizzoli |
| 3.1 | 28/7/2015 | New Section 6 Data Management Tools has been added. Final quality check. | P. Fraternali, A.E. Rizzoli |

# Disclaimer

This document contains confidential information in the form of the SmartH2O project findings, work and products and its use is strictly regulated by the SmartH2O Consortium Agreement and by Contract no. FP7- ICT-619172.

Neither the SmartH2O Consortium nor any of its officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

**The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2013-11) under grant agreement n° 619172.**

The contents of this document are the sole responsibility of the SmartH2O consortium and can in no way be taken to reflect the views of the European Union.

# Table of Contents

# Executive Summary

The main purpose of this deliverable is to design a software repository for storing all types of user related data, from water consumption data, to user psychographic data, down to the user interactions with the SmartH2O platform.

- Section 2 provides a literature review on past residential water end use studies, which were conducted in the last years. Based on the analysis of these water end use studies at the household level, a preliminary set of the main determinants influencing water consumption are identified, eventually determining a set of potentially relevant variables to be included in the SmartH2O database, such as end-user profile data, hydroclimatic data, socio-economic data. This set of variable will be gradually enriched based on the interactions with water utilities and users.
- Section 3 then details the structure of the SmartH2O database, defined in terms of Entity-Relationship models, along with a list of different services allowing the interaction between the users, the smart meter infrastructure, and the SmartH2O database. The database structure has been deliberately kept open and flexible to accommodate additional information coming from the interaction with the water utilities and the end users.
- Section 4 describes the database implementation, in terms of the database storage technology employed, the access endpoints and the data acquisition procedures.
- Section 5 describes the data governance policy that will be adopted in the SmartH2O project.
- Section 6 contains an overall description of the data management tools, more specifically the data assimilation component and the domain modller used to design and create the database.
- An appendix gives the complete source code for generating the SmatH20 database, divided in the principal areas that constitute the data model: the consumer portal the game platform, and the gamification engine database subschemas.

# 1. Introduction

Individual and collective behavioural responses to different water conservation policies acting on the demand side of residential water consumption (the so called Water Demand Management Strategies, WDMS) might significantly vary within the same urban context depending on economic drivers as well as socio-psychological determinants. The SmartH2O project aims at providing water utilities, municipalities, and citizens, with an ICT-enabled platform to design, develop and implement improved WDMS. They will rely on a shared understanding of the water users' behaviour and motivations to reduce water consumption, without compromising the quality of life of the users. SmartH2O builds a bi-directional communication stream between citizens and the water utility: in one direction, user behavioural data are collected by water utilities through smart meters and an online social participation application (social game); in the other, awareness campaigns and price signals are delivered to users through the same app, thus informing them on how to save water and money.

Within the SmartH2O project, Work Package 3 aims at:

- collecting historical and real time water consumption data both at high resolution (i.e., from smart meter infrastructures) and at low resolutions (i.e., billed data);
- identifying water end-use patterns;
- analysing and classifying the consumers' behaviors;
- identifying individual consumer behavioural models;
- developing models of consumers' elasticity to incentives, to awareness campaigns and to social pressure at a single-household level. It is worth mentioning that the final user models should also be able to describe the future consumers' behavior in face of water price changes. The latter is the main goal of Work Package 5 ("*Saving water by dynamic water pricing*"), where econometric models of water demand under new pricing policies will be developed, and eventually integrated with the consumer behavioural models developed in WP3.
- integrating the individual consumer models into a multi-users model exploiting agent-based modeling platforms.

In order to fulfill the WP3's objectives, it is essential to:

- understand which user and household attributes (e.g., number of occupants in a house, garden area, etc.) and exogenous variables (e.g., external temperature, rainfall, etc.) influence water consumption at the household level;
- evaluate the impact of policy actions (awareness campaigns, incentives and social pressure) on the water users' behavior.
- decide how the water utilities taking part in the SmartH2O project (i.e., TWUL and SES) will transfer water consumption data to the SmartH2O platform;
- evaluate how accurate the meter readings should be (in terms of frequency and resolution) to build reliable models of water consumers' behavior.
- understand which actions (e.g., questionnaires, social games) should be taken in order to gather psychographics data on the water consumers.
- organize the gathered information in a database and develop automated procedures to update its content with online meter readings. The developed database should also include data that will be used in WP5 for developing econometric models of water demand under pricing policies.

Starting from a literature review on the residential water end use studies conducted in the last twenty years, relevant insights to face the key steps mentioned above were obtained. Furthermore, taking inspiration from these state-of-the-art residential water end use studies, we identified a preliminary set of data relevant for water users modeling and profiling and,

then, we selected the most appropriate structure to be used in the SmartH2O project. Specifically, the *Platform Model* (which represents the data model on which the components of the SmartH2O platform are founded) has been developed. The Platform Model describes the logical structure of the data processed by the SmartH2O platform. It is defined in terms of an *Entity-Relationship* model which includes and integrates the user data that will be made available by water utilities with additional information about users provided by the *game with a purpose* (GWAP) application developed in WP4.

In the reminder of this deliverable we will describe the various features of the SmartH2O Platform model, which is structured in two main parts, the Consumer Data Model, focussing on the characteristics and features of the water users, and the User Gaming Model, which complements the previous data model to incorporate the gamification components of the SmartH2O Platform model.

# 2. Review of the state of the art

## 2.1 Users' water consumption and psychographics datasets

In the last two decades, several residential water end use studies have been conducted. Among these studies, we mention:

- The *Residential End Uses of Water Study*[1] (REUWS), funded by the *American Water Works Association Research Foundation* (AWWARF) from 1996 to 1999.
- The *Water End Use and Efficiency Project*[2] (WEEP), funded by the Building Research Levy, New Zealand, from 2005 to 2007.
- The *California Single-Family Water Use Efficiency Study*[3], funded by the *California Department of Water Resources*, from 2005 to 2010.
- *Albuquerque Single-family Water Use Efficiency and Retrofit Study*[4], funded by the *American Recovery and Reinvestment Act* (ARRA) in 2009.
- The *South East Queensland Residential End Use Study*[5] (SEQREUS), funded by the *Queensland State Government*, Australia, from 2009 to 2011.
- The *H2ome smart* project[6], funded by the *Water Corporation,* Western Australia, from November 2010 to February 2012.

The common goals of these residential water end use studies were:

- disaggregating water flow data into different water end use categories to design effective water saving campaigns;
- identifying the main determinants of residential water consumption;
- classifying households for water demand forecasting;
- profiling water users to determinate potential water saving actions within each profiled group of users;
- providing feedback to the users on water consumption.

Extensive databases on users' water consumption and consumers' behavior were developed throughout these studies. Data on household water consumption were gathered through high resolution (i.e. up to 72 pulses per liter and 5 – 10 seconds as data logging frequency) smart meters. Psychographic data about water users and information on consumers' behaviour were gathered through household auditing, questionnaires, and self-reported diaries (filled out by the users registering the use of water-consuming appliances/fixtures during monitored days). In the following paragraphs, we provide a brief discussion on the data collected during the aforementioned residential water end use studies.

---

[1] P. W. Mayer and W. B. DeOreo, *Residential end uses of Water*, AWWA Research Foundation and American Water Works Association, 1999. Available online at: http://www.aquacraft.com/node/56

[2] M. Heinrich, *Water End Use and Efficiency Project*, 2007. Available online at: http://www.branz.co.nz/cms_show_download.php?id=9bf916e031023c9323d5abe093a02a0b0741cc9e

[3] *California Single-Family Water Use Efficiency Study*, Aquacraft Inc., 2011. Available online at: http://www.aquacraft.com/node/63

[4] *Albuquerque Single-family Water Use Efficiency and Retrofit Study*, Aquacraft Inc., 2011. Available online at: http://www.aquacraft.com/node/71

[5] C. Beal and R. Stewart, *South East Queensland Residential End Use* Study-*Final Report*, 2011. Available online at: http://www.urbanwateralliance.org.au/publications/UWSRA-tr47.pdf

[6] M. Anda, J. Brennan and E. Paskett, *Behaviour change programs for water efficiency: Findings from North West and Metropolitan Residential Programs in Western Australia*. In: IWA World Water Congress & Exhibition, September, Busan, Korea, 2012.

*REUWS project (1996-1999)*

The main objectives of the *REUWS* project were to figure out where, when and how water is used in single-family houses in North America, by disaggregating water flow data into different end uses categories (e.g., toilet flush, shower, dish washer, etc.). Twelve locations in North America were analyzed (i.e., Boulder, Colorado; Denver, Colorado; Eugene, Oregon; Seattle, Washington; San Diego, California; Tampa, Florida; Phoenix, Arizona; Tempe and Scottsdale, Arizona; the Regional Municipality of Waterloo, Ontario; Walnut Valley Water District, California; Las Virgenes Municipal Water District, California; and Lompoc, California). The gathered data include:

- Historic billing records from 12,000 single-family detached residential accounts (1,000 per study site);
- Household level information, obtained through a detailed mail survey sent to each of the 12,000 monitored households. The survey included questions about the number and the type of water-using fixtures present in the residence, landscape characteristics, irrigation methods and habits, adopted water conservation actions, type of residence, household demography, size and economic value of the house, household income, etc. The mail survey was completed by approximately 6,000 households.
- Data on the end uses of water, collected for approximately four weeks from a total of 1,188 households (approximately 100 per study site). Water consumption for various end uses was measured through compact data loggers and a PC-based flow trace analysis software. A flow trace is a record of flow through a residential water meter, recorded at 10 seconds intervals, which provides sufficient resolution to identify the patterns of specific fixtures within the household.
- Daily weather data (e.g., max temperature and total precipitation per day) obtained for each individual household from local weather measurement stations.

Further details on the structure of the REUWS database can be found in Appendix C of the report *Residential end uses of Water*, authored by Mayer and DeOreo, published by the AWWA Research Foundation and American Water Works Association in 1999.

*WEEP (2005-2007)*

The WEEP project aimed at developing automated methodologies for monitoring the end uses of water in residential buildings. Twelve houses on the Kapiti Coast (New Zealand) were monitored for a period of approximately six months, i.e., from mid-July to mid-October 2006 and from mid-November 2006 to end of February 2007. Two separate periods have been monitored to capture seasonal variations. The data gathered throughout this study include:

- Water consumption data collected at a 10-second interval from high resolution (approximately 30ppL) smart meters.
- Measurements of the signature trace of each fixture/appliance. In order to collect this information, each appliance was turned on for at least 1 minute, while all other appliances were turned off. The maximum flow rates of each tap were also measured, using a conventional bucket and stop watch technique.
- Psychographic data of each household and information on users' behavior, obtained through a questionnaire sent to the monitored users. Such a questionnaire is reported in Appendix A of the report: *Water End Use and Efficiency Project-Final Report* (2007), by M. Helnrich.

*California Single-Family Water Use Efficiency Study (2005-2010)*

The main goals of the *California Single-Family Water Use Efficiency Study* were:

- to assess the efficiency of water use (and then to estimate remaining conservation potential) in single-family homes in the State of California;

- to provide information on the rate of adoption of high-efficiency fixtures and appliances by California homeowners;
- to provide information that can be used by California water agencies to update their Urban Water Management Plans;

A sample of over 732 single-family households across ten water agencies throughout the State of California was monitored between November 2006 and August 2008. Data collected from this study include:

- Two-week water consumption from each monitored household. Flow trace data were collected at a 10-second interval from smart water meters installed in each house. Flow trace data were disaggregated into end uses using the proprietary *Trace Wizard®* software (developed by Aquacraft, Inc.).
- Information about water conservation programs employed by the 10 water agencies participating to the study obtained through surveys sent to the water agencies.
- Physical, demographic and attitude information on the costumers participating to the study, obtained through surveys sent to the costumers.
- The irrigated area for each of the study household (analysed according to the plant type and the irrigated area), obtained by ortho-rectified aerial photos provided by the water agencies and through geographic information system (GIS) technology.

*Albuquerque Single-family Water Use Efficiency and Retrofit Study (2009)*

The goal of the *Albuquerque Single-family Water Use Efficiency and Retrofit Study* was to obtain a detailed analysis on the indoor and outdoor water use patterns of a random sample of single-family homes in the service area of the Water Authority *Albuquerque Bernalillo County Water Utility Authority* (operating in the State of New Mexico) and, at the same time, to determine the percentage of homes that meet specific criteria for high efficiency fixtures and appliances.

A total of 3000 homes were sampled from the Water Authority's billing database for survey mailing, and annual/seasonal water use analysis. In order to examine the impact of the Authority rebate program on water use, one half of the survey group (1500 customers) was randomly selected among those customers who did not receive any rebates from the Water Authority, and the other half was selected from customers who received either an indoor or an outdoor rebate, or both. From returned surveys, a random sample of 240 household was selected for data logging in order to obtain detailed end-use information. A second component of the study was a retrofit analysis on a group of 29 homes chosen from the baseline group. This retrofit group had its fixtures and appliances upgraded to high efficiency devices and their water use was measured afterwards to determine the potential savings from the program. The survey can be found in Appendix A of the report *Albuquerque Single-family Water Use Efficiency and Retrofit Study* (2011) prepared by Aquacraft. The data gathered throughout this study include:

- 10-second flow trace data collected from the main water meters serving study homes. The flow of water was recorded for a two-week period. Other two weeks of flow trace data were collected once the retrofits were complete.
- Disaggregation of the water flow trace into individual water use events (the disaggregation has been performed through the software package *Trace Wizard®*)
- Local climate data (measured from local weather stations).
- Efficiency of the irrigation system and average water needs of the plants in a landscape (estimated data).
- Theoretical Irrigation Requirement (TIR), which measures the amount of water needed to maintain a reasonable landscape in an urban environment. The TIR of a landscape has been estimated based on the characteristics of plant type, microclimate, density, and efficiency of the irrigation system of each sub-area composing the landscape.
- Landscape Area of the residential sites was estimated using the high-resolution aerial images made available from the City of Albuquerque. The detail provided by these images

generally made it possible to differentiate between turf areas, shrub borders, deciduous and coniferous trees, low-water use planting, and non-irrigated areas. Ground observation was used to confirm (or update) the findings from the aerial images.

*SEQREUS project (2009-2011)*

The objectives of SEQREUS project were calculating household and per-capita end-uses consumption rates, revealing key determinants of water end-use demand, studying diurnal demand patterns at an end-use level and assessing the influence of water-efficient appliances.
Data collected from this study include:

- Water consumption data collected from 252 detached households in four interconnected cities (i.e., Brisbane, Gold Coast, Ipswich and Sunshine Coast) located in the South East Queensland region, in Australia. Data are collected from 3 periods: from 14$^{th}$ of June 2010 to 28$^{th}$ of June 2010; between the 1$^{st}$ of December 2010 to the 21$^{st}$ of February 2011; from the 1$^{st}$ of June 2011 to the 15$^{th}$ of June 2011. Water flows were measured by smart meters with a resolution of 72 pulses/litre. The smart meters were connected to data loggers, which were programmed to record pulse counts every five seconds. Data were wirelessly transferred to a central computer and stored in a database.
- demographics and socio-economic variables for each of the 252 metered households. They include: number of occupants, age of occupants, annual income, and education level.
- Water flow trace patterns of each appliance/fixture in each metered household. This patterns are identified through stock surveys and self-reported water diaries filled out by the householders over a seven-day period. The proprietary *Trace Wizard®* software was used in conjunction with stock surveys and water diaries to analyse and disaggregate consumption into the following end use event categories: toilets, taps, leaks, irrigation, shower, washing machine, bathtub and dishwasher.

*H2OME SMART project (2010-2012)*

The H2HOME SMART project aimed at empowering residents to make practical and sustainable behavioral changes in their water use by providing personalized feedbacks (via telephone conversations or letters). The projected involved 9 towns of the Pilbara and Kimberley Regions of Western Australia, engaging 4,338 households.
A final evaluation of the program estimated savings equal to 6.9% of the expected consumption in 2011. Also, the program included a residential retrofit campaign to pursue additional savings.
Data collected during this study include:

- Low resolution, water consumption billed data were collected from 6 meter reads rounds: Jan-Feb 2011, march 2011, July-Aug 2011, Aug-Sept 2011, Oct-Nov 2011 and Jan-Feb 2012. The first five rounds of meter reads included both participants and non-participants, while only participants were considered for reading 6. A total of 16,383 meter read were obtained after the first round, then this number decreased between rounds 2 and 5, but it was always above the targeted value of 12,500 reads. Reading 6 was obtained only for 3,681 households (only participating households had meter reads).
- For each household, demographics and socio-economic data characterizing the occupants were also collected, including, among others, household type, account type, household responsibility, number of occupants, number of toilets, garden area, irrigation techniques, pool presence.

Data regarding the retrofit campaign of the project (May 2011 – February 2012) report about a total of more than 11,000 retrofit items and installations occurred at 2,286 eligible household before the end of February 2012.

## 2.2 Meteorological databases

Exogenous factors, such as external temperature, precipitation and/or drought conditions potentially influence residential water use. Numerous studies, aiming at analyzing the effect of climate variables on water consumption, have been conducted especially in North America. For instance, in the *REUWS*[1] project, it was found that, across 12 cities in North America, net evapotranspiration explained 59% of the variation in outdoor water use. Guhathakurta et *al.*[7] analyzed the spatial effects of June nighttime temperature on residential water use in Phoenix. They developed a statistical model indicating that an increase of 1°C results in an increase in household water consumption of 4.61 $m^3$ annually. The above considerations point out the need for considering local meteorological data (in particular, daily temperature, humidity and precipitation) for accurately modeling and profiling water users.

On the other hand, it has to considered that the effect of varying climatic conditions on residential water consumption might be very different according to the efficiency of water-using equipment; also automatic temperature and humidity sensing devices can greatly change the amount of water required for landscaping, even if they are not yet widespread for household use.

In order to properly account for meteorological conditions both historical and near real-time data are needed. Indeed, historical data are necessary for modeling and understanding the consumer behavior and for comparing different types of users. Near real-time data are necessary to validate and update the consumer model, and to make a short-term prediction of the water consumption.

Climate data can be collected from ground operation systems and upper-air systems (e.g., weather balloons, weather radars, aircraft observations, satellite observations). Several climate datasets are available in the literature, such as:

- The *Met Office Integrated Data Archive System (MIDAS) Land and Marine Surface Stations Data*[8]. This dataset contains land surface measurements as reported by stations in the U.K. and globally. Available measurements include daily and hourly weather observations, hourly wind parameters, max and min air temperatures, daily, hourly rain measurements, soil temperature parameters, sunshine duration and radiation measurements from 1853 to date. The *MIDAS* data are restricted and they can be used for free for academic research. In order to obtain the *MIDAS* dataset, an application has to be submitted to the British Atmospheric Data Center.
- The *Data Warehouse* (DWH) of MeteoSwiss[9]. Measurements of temperature, humidity, precipitation collected every 10 minutes from almost 120 surface stations located over Switzerland. The data can be requested to MeteoSwiss, subject to a fee.
- The *Global Summary Of the Day* (GSOD), which contains climate data from more than 9,000 stations located around the world. The data are obtained from the U.S. Air Force Climatology Center, they are updated approximately daily, they can be download for free from the GSOD's website[10] and can be used for non-commercial purposes. The daily variables included in the GSOD's dataset are, among others: minimum, maximum and average temperature, precipitation amount and snow depth.

---

7 S. Guhathakurta, S. Gaber, P. Gober, *Impact of urban heat islands on residential water use: The case study of metropolitan Phoenix*. North American Regional Science Council Annual Meeting, Las Vegas, Nevada, USA, 2005.

8 http://badc.nerc.ac.uk/view/badc.nerc.ac.uk__ATOM__dataent_ukmo-midas

9 http://www.meteoswiss.admin.ch/home/research-and-cooperation/international-cooperation/gcos/national-climate-observation.html

10 http://www7.ncdc.noaa.gov/CDO/cdoselect.cmd?datasetabbv=GSOD&countryabbv=&georegionabbv=

Since the *GSOD*'s dataset contains updated daily measurements from stations located in Ticino (Switzerland) and in London (United Kingdom), i.e. the two case studies of the SmartH2O project, we decided to use this dataset throughout the SmartH2O project for the purpose of user modeling and profiling.

# 3. Database structure

This section shows the structure of the database of SmartH2O, called *SmartH2O db* hereafter, which is currently under development. As real data become available, the database structure can be expanded to obtain a finer and richer structure.

## 3.1 Platform data model description

In this section, we define the data model of the SmartH2O platform.

The data model describes the logical structure of the data processed by the various components of the SmartH2O platform in terms of entities and relationships following the Entity-Relationship model. It includes and integrates the user data made available by the water utilities and additional information about users provided by the game with a purpose (GWAP) developed in WP4.

The Platform data model comprises two components:

- the **Consumer Data Model:** described in Section 3.3, contains the set of entities and relationships that express knowledge about user data made available by the water utilities (smart metered or surveyed).
- the **User Gaming Model:** described in Section 3.4, focuses on a specific class of actions, which are deployed in the form of a gamified application or of a GWAP and expresses the engagement and rewarding mechanisms typical of gaming.

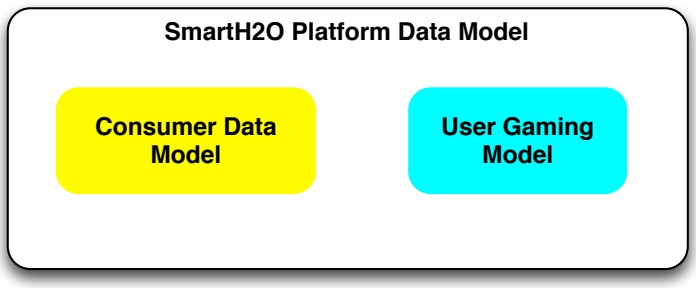


**Figure 1. The SmartH2O Platform Data Model**

## 3.2 Data requirements

In the following tables (Tables 1-3) we list the data related to users, houses, billing prices and policies with their dimensional unit and their justification in the SmartH2O scenario, that have been considered during the development of the SmartH2O db. The data will be used during the project to infer information about users' profiles and to estimate econometric models of water demand (main goal of WP5).

In **Table 1** "high-priority" building data are listed. These data will be considered in the Consumer Data Model of SmartH2O presented in Section 3.3.

**Table 1: High-priority building data**

| NAME | DESCRIPTION | UNIT | JUSTIFICATION: the variable is necessary to deliver the final Water Demand Management (WDM) strategy and more particularly | |
|---|---|---|---|---|
| | | | **…to profile users** | **…to estimate econometric models of water demand** |
| Number of occupants | Number of house occupants | [-] | - To evaluate the average water consumption per capita<br>- To classify households for water demand forecasting<br>- For user modeling and profiling | - To disentangle the impact of price and other policies from structural determinants of water consumption<br>- To have per capita water consumption |
| Household location | Zip code/water reading group ID (Town, suburb)<br>**Full postcode is fine** | [-] | - To develop an agent-based model<br>- To cluster consumption data and other psycho-demographic data based on location (spatial analysis) | - To model the impact of social ties (i.e. a possible driver of water consumption)<br>- To disentangle the impact of price and other policies from the role of socio-economic determinants (e.g. rural vs. urban) and exogenous drivers (e.g. climate) |
| Residency type | Household category (e.g. flat, single house, etc…) | [-] | - To classify households for water demand modeling and forecasting<br>- To cluster similar house types and compare the features of their inhabitants and their water consumption<br>- For user modeling and profiling | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Water consuming devices presence | Binary variable indicating the presence of water consuming devices (e.g. washing machine, shower, faucet, dishwasher, etc…) | [binary] | - To disaggregate water flow data into different water end use categories<br>- For user modeling and profiling<br>- End use focused Water Demand Management (WDM) strategies delivery | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Garden area | Area of the garden, if present, zero otherwise | [$m^2$] | - Critical attribute for disaggregation and profiling<br>- WDM strategy delivery<br>- To classify households for water demand forecasting<br>- Its potentially one of the most contributing factors to residential water consumption | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Income rate | Gross pre-tax yearly income of the whole | [£/year (or | - To understand and model its link with water consumption | - To disentangle the impact of price from |

| | | | | |
|---|---|---|---|---|
| | household | month)] | - To develop accurate agent-based models, in which we will verify how much the income rate influences the acceptance of awareness campaigns. | wealth effects |
| Billing price | Monthly service charge (£) and volume charge (£/L)<br>**Panel data, i.e., sample of same households observed overtime** | [£], [£/L] | - To disentangle the price effect from the effects of other drivers<br>- To find out if and how price level and tariff structure have driven consumption behaviour | - To find out if and how price level and tariff structure have driven consumption behaviour |
| User type | Only if users other than households included (Household; Commercial or small business; Industrial) | [-] | - To cluster consumption and end uses according to the type of user<br>- To develop accurate agent-based models. Indeed, water consumption awareness depends whether the consumer will pay the bill or not, as well as whether he/she able to periodically see his/her water consumptions (i.e., if he/she an employer of a company, etc.) | - To disentangle the impact of price and other policies from structural determinants of water consumption (i.e. consumption practices and rates are different between residential and business users) |

In **Table 2** exogenous data considered in the SmartH2O db are listed.

**Table 2: Exogenous Data**

| **NAME** | **DESCRIPTION** | **UNIT** | **JUSTIFICATION: the variable is necessary to deliver the final WDM strategy and more particularly** | |
|---|---|---|---|---|
| | | | **…to profile users** | **…to estimate econometric models of water demand** |
| Rainfall | Time series of rainfall data.<br>Needed at least one year, in order to consider the seasonality | [mm/day] | - For user modeling and profiling, as this variable influences water consumption. | - To disentangle the impact of price and other policies from other exogenous drivers |
| Temperature | Time series of temperature data.<br>Needed at least one year, in order to consider the seasonality | [°C] | - For user modeling and profiling, as this variable influences water consumption. | - To disentangle the impact of price and other policies from other exogenous drivers |

In Table 3 other information about users and buildings are listed. These data will be considered in future extensions of the Consumer Data Model of SmartH2O presented in Section 3.3.

**Table 3. Other low-priority data about users and households**

| NAME | DESCRIPTION | UNIT | JUSTIFICATION: the variable is necessary to deliver the final WDM strategy and more particularly | |
|---|---|---|---|---|
| | | | **…to profile users** | **…to estimate econometric models of water demand** |
| Occupants age | Age of house occupants | [-] | - For user modeling and profiling, and to understand if this variable influences water consumption.<br>- WDM strategy delivery | - To disentangle the impact of price and other policies from the role of socio-economic determinants |
| Years of occupancy | Number of years the house is being occupied by the same users | [-] | - For user modeling and profiling, and to understand if this variable influences water consumption.<br>- WDM strategy delivery | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| House age | Number of years since the house was built | [-] | - For user modeling and profiling. Indeed, old house might not have water-efficient devices such as flush toilets and showerheads. Thus, WDM strategies should be targeted to replace the non-efficient devices. Furthermore, old house can have leaking water pipes. | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| House size | Area of the house (cadastral area) | [$m^2$] | - For user modeling and profiling<br>- To classify households for water demand forecasting<br>- To see whether it is related to other important factors (e.g. number of occupants and number of toilets) | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Household responsibility | Type of house ownership (e.g. owned, rent, house provided by employer, etc…) | [-] | - To develop accurate agent-based models. Indeed, water consumption awareness depends if the consumer will pay the bill or not (i.e., if he/she is a tenant and his/her lease does not depend on water consumption)<br>- WDM strategy delivery | - To disentangle the impact of price and other policies from the role of other determinants |
| Rural | Census classification or share of municipal rural area – if household location is not | [Yes/No] or [%] | - To classify households for water demand forecasting | - To disentangle the impact of price and other policies from the role of other determinants |

| | | | | |
|---|---|---|---|---|
| | sufficient | | | |
| Density | Population density – if household location is not sufficient | [1,000 inhabitants / $km^2$] | - To see whether it is related to other important factors (e.g. type of house) | - To disentangle the impact of price and other policies from the role of other determinants |
| Second | Used only for holidays or weekends | [binary] | - To classify households for water demand forecasting | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Watering method | Watering method technique | [-] | - To understand how technologies and watering methods influence water consumption<br>- WDM strategy delivery | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Watering time | Watering time | [min/day] or [min/week] | - WDM strategy delivery (mostly educational) | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Pool presence | Binary variable | | - To classify households for water demand forecasting<br>- WDM strategy delivery<br>- To disaggregate water flow data into different water end use categories<br>- Its potentially one of the most contributing factors to residential water consumption | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Pool cover presence | Binary variable | [-] | - WDM strategy delivery. Indeed, the presence of a pool cover it's important to keep the pool clean and prevent water loss through evaporation | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Water consuming devices type/efficiency level | Any qualitative/quantitative data about water consuming devices class, features and efficiency | It depends on the available data | - WDM strategy delivery<br>- To disaggregate water flow data into different water end use categories | - To disentangle the impact of price and other policies from structural determinants of water consumption<br>- To understand if the existence or adoption of any water consuming devices influence water consumption |
| Number of toilets | Number of toilets in the | [-] | - To disaggregate water flow data into different water end | - To disentangle the impact of price and other policies from |

| | | | | |
|---|---|---|---|---|
| | house | | use categories | structural determinants of water consumption |
| Household shower consumption per day | Amount of water used for showering | [L/hh/day] | - WDM strategy delivery<br>- To classify households for water demand forecasting<br>- To disaggregate water flow data into different water end use categories | - To disaggregate water consumption in the home into several indoor fixtures e.g. shower consumption, so that we can explore the role of dynamic pricing policies such as seasonal or peak load pricing. |
| Showering time | Estimated showering time | [min/day] | - WDM strategy delivery<br>- To classify households for water demand forecasting | - To disentangle the impact of price and other policies from structural determinants of water consumption |
| Household clothes washer consumption per day | Estimated water consumption devices rate | [L/hh/day] | - WDM strategy delivery<br>- To disaggregate water flow data into different water end use categories | - To disaggregate water consumption in the home into several indoor fixtures e.g. clothes washer consumption so that we can explore the role of dynamic pricing policies such as seasonal or peak load pricing. |
| Household tap consumption per day | Estimated water consumption rate | [L/hh/day] | - WDM strategy delivery<br>- To disaggregate water flow data into different water end use categories | - To disaggregate water consumption in the home into several indoor fixtures e.g. tap consumption so that we can explore the role of dynamic pricing policies such as seasonal or peak load pricing. |
| Household toilet consumption per day | Estimated water consumption rate | Estimated water consuming devices rate | - WDM strategy delivery<br>- To disaggregate water flow data into different water end use categories | - To disaggregate water consumption in the home into several indoor fixtures e.g. toilet consumption so that we can explore the role of dynamic pricing policies such as seasonal or peak load pricing. |
| Well | Well or other source in the house – replacing device (partially replacing water service) | [yes/no] | - To understand which alternatives sources of water each household has | - To understand which alternatives sources of water each household has |
| Users' education level | User's education level, e.g. high school degree | [-] | - To understand and model its link with water consumption<br>- Propose WDM strategies based on the users' | - To disentangle the impact of price and other policies from the role of socio-economic determinants |

| | | | | |
|---|---|---|---|---|
| | | | education level | |
| Users' perceived environmental commitment | Personal environmental commitment perceived by the user.<br>Qualitative class, e.g. «high, medium, low» | [-] | - WDM strategy delivery<br>- Compare the expected consumption with the actual user attitude | - To disentangle the impact of price and other policies from the role of socio-economic determinants<br>- To understand if and how individuals' environmental attitudes and behaviors influence water consumption |

## 3.3 Consumer Data Model

The Consumer Data Model comprises the set of entities and relationships that express knowledge about user data made available by the water utilities (smart metered or billed). This knowledge, which is used for analysis purpose, can be automatically produced by smart meters, obtained from on-line bills, or manually produced by users that interact with the GWAP developed in WP4.

In essence, the Consumer Data Model of SmartH2O is organized into a database designed following the Entity-Relationship model.

### *3.3.1 Description of the main entities of the Consumer data model*

**Household**: it identifies the concept of household (a.k.a. "family"). Each household has an identification [Oid], an [UtilityID] linking the house to the water utility, the size [Household Size], a flag stating if the Head of household is either a tenant or an owner [Ownership], the number of occupants [Number Occupants], the presence of pets, if any, [Number Pets], the area of the garden (if any) [Household Garden Area], the volume of the pool (if any) [Household Pool Volume], a flag stating if the house is used only for holidays or weekends [Second], a flag stating if the household discloses the geo-location to other users [Visible], a flag stating if the household discloses household information to other users [Public].

Each Household could have up to *n* Devices (Device Class). Each **Device Class** device has an identification [Oid], the name of the device [Name] and the number of pieces of that device present in the considered house [Number].

**Device Consumption**: it identifies the consumption data, *disaggregated by Device*, result of models computation. Each Device Consumption has an identification [Oid], a given interval [Start Date][End Date] ant the consumption value [Device Consumption].

**Bill**: each bill is identified by the account number [Account Number], the date [Bill Date] and the company [Company] which invoiced the bill. Moreover, for each bill we store the charge for water supply [Volume Charge], the charge for service supply [Service Charge] and the currency of that bill [Currency]. Each bill is in association with the Household the bill is referring to and has some Billing Price composition.

**Billing Price**: for each month [Month], year [Year] and company [Company], it stores the monthly service charge [Monthly Service Charge] and volume charge [Monthly Volume Charge].

**Household Consumption**: identifies the consumption data, *disaggregated by Household*. If no disaggregation is computed, it will store the original consumption data coming from smart

meter readings. Each Household Consumption has an identification [Oid], the given interval [Start Date][End Date], and the computed consumption value [Consumption].

Each Household could have *n* **NeutralUsers** (a.k.a. “family members”). Each Neutral User inherits an identification attribute [Oid] from the **User** entity, the authentication information [Username] and [Password], the email [Email], the firstname [Firstname] and lastname [Lastname] and the birthdate [Birthdate]. Moreover, for each **Neutral User** we store the registration date [Registration Date], the name of his/her role in the family [Family Role], his/her educational level [Educational Level], the economic level [Income Rate], the money system adopted by the user [Currency], a flag stating if the user discloses personal information to other users [Public], the user language [Language], the temperature unit [Temperature Unit] and the length unit adopted by the user.

The data model implements the Role Based Access Control (RBAC): Users are clustered in Groups, which represent the various classes of users. Each **Group** has an identification attribute [Oid] and a name [GroupName]. Groups are connected to Modules, which represent the interfaces to the SmartH2O resources that the class of users is entitled to access.

Each **Module** has an identification [Oid], a name [ModuleName] and the name of the module domain [ModuleDomainName].

In order to provide more appropriate and targeted incentives, Neutral Users are grouped into consumer segments. Each **Consumer Segment** is identified by a unique id [Oid], a name [Name] and a description [Description]. A segment of users is characterized by a set of features. Each **Feature** is identified by a unique id [Oid], a type [Type] and a level [Level] (e.g. Consumption Average: medium, Environmental Commitment: high).

**Media Asset**: each media object provided to users is identified by a unique id [Oid], a title [Tile], a description [Description], the author [Author], the duration of the video [Duration] and the URL of the media object [Media].

Some Tips are provided to users. Each **Tip** is identified by a unique id [Oid], a name [Name] and the text content divided into a header [Header] and a body [Body].

Users can be notified about possible leaks or bad water quality through alerts. Each **Alert** is identified by a unique id [Oid], a type [Type] (e.g. Water Quality Alert, Leakage Alert, Shortage alert), a level [Level] (e.g. low, medium, high). When a new alert is inserted, the current date [Date] is stored in order to keep track of the progress of a particular type of alert and to record past critical situations.

An Alert can be associated to a Mail, in order to directly notify the user. Each **Mail** is identified by a unique id [Oid], a description [Description], the subject of the email [Subject], the body of the email [Body ]and the language [Language].

**Building:** it identifies the physical building, containing one or more Households. Each building has an identification [Oid], an address [Address], the area of the garden (if any) [Building Garden Area], a description of the type of residence [Residence Type], the size [Building Size], the number of years since the house was built [Age], the volume of the pool (if any) [Building Pool Volume].

Each building could be metered by one**Smart Meter**.

**Meter Reading** stores the readings and each of them has an identification [Oid], the timestamp [Reading Date Time], the company [Company] and the actual reading [Total Consumption].

Each building is also associated to the District where it is located. Each **District** has an identification attribute [Oid], a Zip code [Zip Code], the name of the country [Country] and the city it belongs to [City] and the name of the district [Name].

**Weather Condition**: the entity stores, for a given interval [Start Date][End Date], the quantity of rain [Rain Fall] and the Average Temperature [Average Temperature] in a certain District.

**Unit Of Measurement**: stores the information needed to perform conversions. Each conversion is applied to a given physical quantity [Physical Quantity] and is characterized by a unique id [Oid], the primary [Primary Unit] and secondary [Secondary Unit] unit of measure and the coefficient to be applied in order to perform the conversion.

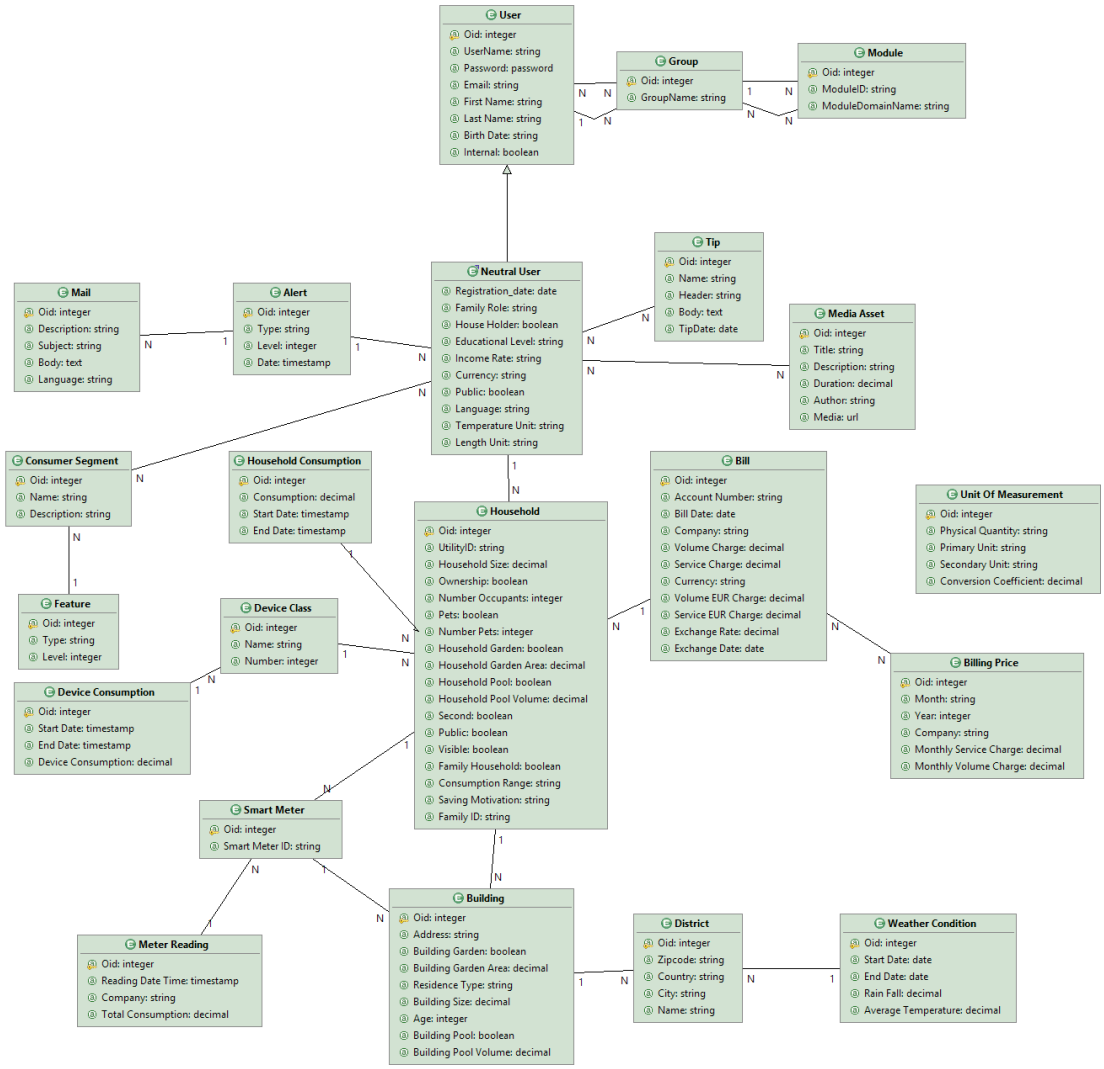


**Figure 2. The Consumer Data Model of SmartH2O**

## 3.4 User Gaming Model

The User Gaming Model comprises the set of entities and relationships that express knowledge about user data made available by the GWAP developed in WP4.

Ii is logically divided into two sub-schemas:

- The **Gamification engine subschema data model** describes the entities and relationships necessary to represent the users of business applications that are extended with gamification features.
- The **Game platform subschema data model**, which adds more specific entities and

relationships describing the data requirements for players of the digital games.

***Gamification Engine Data Model***

The **Gamification Engine Data Model** comprises the set of entities and relationships that express knowledge about user data produced and consumed by the Advanced Gamified Customer Portal.

The schema in Figure 3. The Gamification Model shows the implemented gamification engine database. The following entities have been considered:

**Community User**s: the entity is a specialization of **User** and contains all the attributes that identify the user as a member of a community (like credits, bio information, …).

**Gamified Application:** this table contains information about applications that call the gamification engine.

**Action Type:** the entity contains the dictionary of the actions of the gamification engine. The attribute values of an action are the specific features of the considered action.

**Action Instance:** the entity stores all the action instances performed by a user.

**Badge Type:** the entity contains the dictionary of the badges that a user can acquire.

**Badge Instance:** the entity contains all the badge instances acquired by the user.

**Reward Type:** the entity contains the dictionary of the rewards.

**Reward Instance:** the entity contains the instances of the rewards acquired by the users.

**Text Mail:** the entity contains information about the notification to send to users after a particular event in the gamification engine (e.g. a user gains a badge).

**Notification**: this entity contains the notification sent to users.

**Thematic Area**: this entity contains the thematic areas to organize actions and badges according to topics. Each thematic area is identified by a unique id and a name.

**Game Result**: this entity contains the possible outcomes of games that need to be converted into credits. Each game result is identified by a unique identifier, a title (e.g. New level reached) and optionally by a score, a level and the current available lives. Each game result is mapped to an Action Type and, according to the game results attributes (score, level, lives) the game result is converted into credits.

**Game Points Converter**: each conversion is identified by a unique id, the game to which the conversion rule is applied, and the customizable formula which will take the attributes as inputs (score, level, lives) and will produce credits amount as output.

**Alliance**: this entity contains the coalitions created among competitor users. Each Alliance is identified by a unique id, a start date and an end date.

**Goal**: this entity contains the consumption goals assigned to users. Each goal is identified by a unique id, a title, a consumption value, and optionally the completion date. A goal can be assigned to a given user or to an alliance of users. Goal can be associated to a Badge Type, obtained by the user when the consumption goal is achieved.

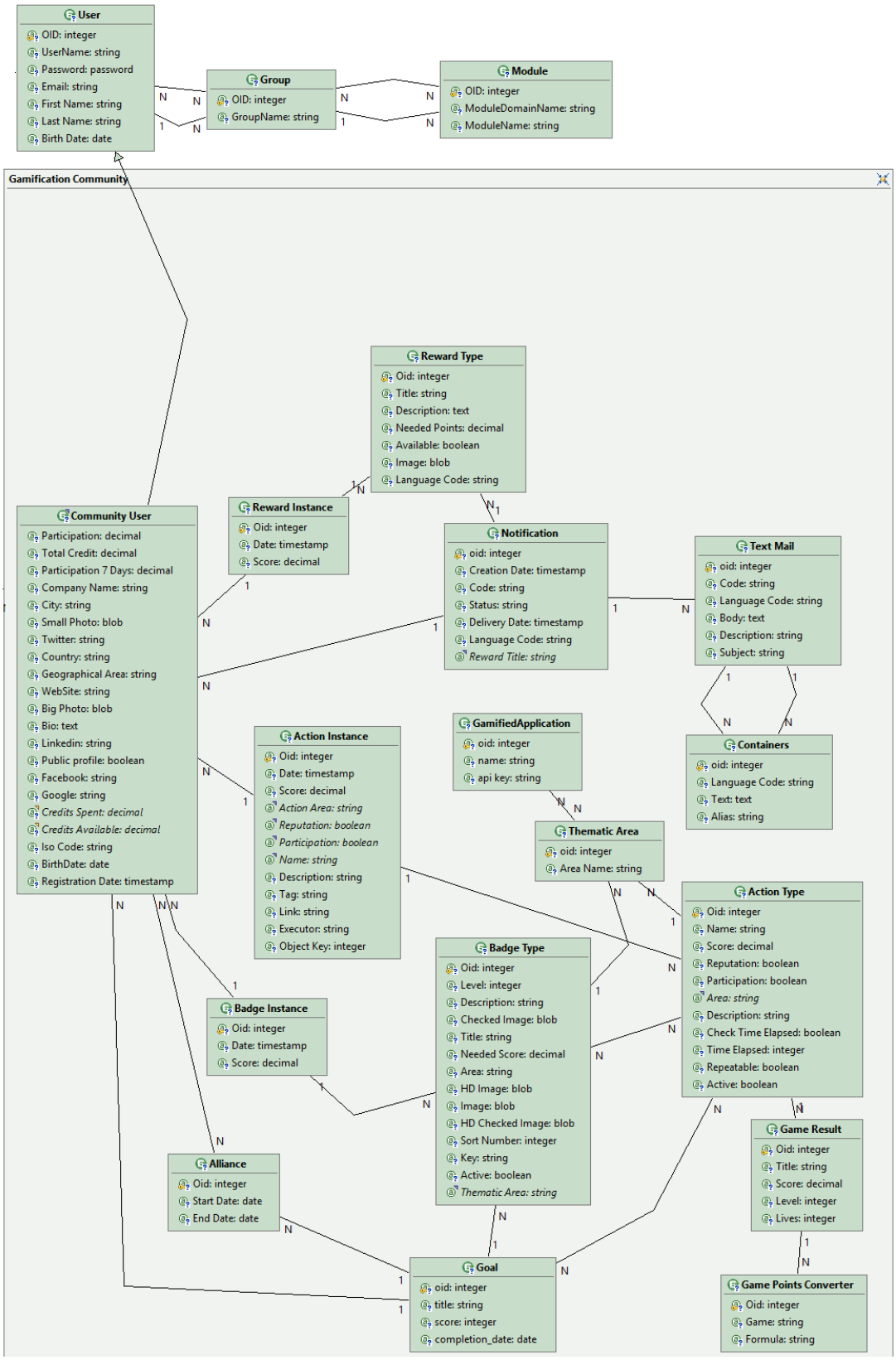


**Figure 3. The Gamification Model**

### *3.4.2 Game Platform Data Model*

The Game Platform data models expands the gamification engine data model with the representation of additional entities and relationships that capture the essential data about the users who play with the SmartH2O digital games.

The Entity Relationship Schema is represented in Figure 4. Game platform data model.

**Game** is the core entity: the *Mode* attribute represents the gameplay modes (e.g. Single Player, Multi Player, Cooperative), while the *Genre* attribute identifies its genre (e.g. Puzzle, Educational). Each game is also characterized by a *Title*, a *Theme* and the *Minimum/Maximum number of players*.

An **Achievement** has an *Icon*, which describes it in a visual way, a *Category* that specifies the task (Instructor, Grinder), an attribute *PointsGiven*, which contains the amount of points to be granted, and a Boolean attribute *OfTheDay* defining whether the achievement has to be completed on a specific day in order to obtain virtual goods, more points, or increased levels.

The **Player** entity accommodates game-specific personal and social features. *Avatar* and *Nickname* allow the user to be recognizable by using a custom image or a unique fictional name, while *Player Type*, *Player Level* and *Experience Points* convey player progress. *Reputation* in online gaming communities is fundamental and distinctive feature of any player; being able to recognize wheter a player is bad mannered, prone to cheating, unpleasant to play with is of utterly importance to assure a satisfying gaming experience for the user of an entertainment platform; it is usually measured as an integer number ranging from 0 to 5.

The model describes also the game-relevant statistics (**GameStats**): the proficiency and the experience of a player in a given game are represented by aggregating in a compact way such indicators as points gathered and hours spent playing.

**GameBadges** represent the achievements that have been unlocked by a player. The *CompletionPercentage* field shows how much the player has already achieved in a specific task. *StartDate* and *EndDate* record the dates in which the player has started to work on the achievement's goals and the date in which he has obtained it. The *TrialsN* attribute tracks how many times the user tried to fulfill the achievement.

A **GamePlayAction** of a player, associated with a specific **Gameplay**, records the *StartDate* and *EndDate* of the gaming session and the actual actions performed by the player on that specific time frame and the *Role* defines which are the allowed actions in the game for the role associated to a player.

In order to store questions and answers required by the Drop!TheQuestion trivia game, **Question** and **Answer** entities have been provided. **QuestionInstance** keeps track of players game play information related to the specific quiz game.

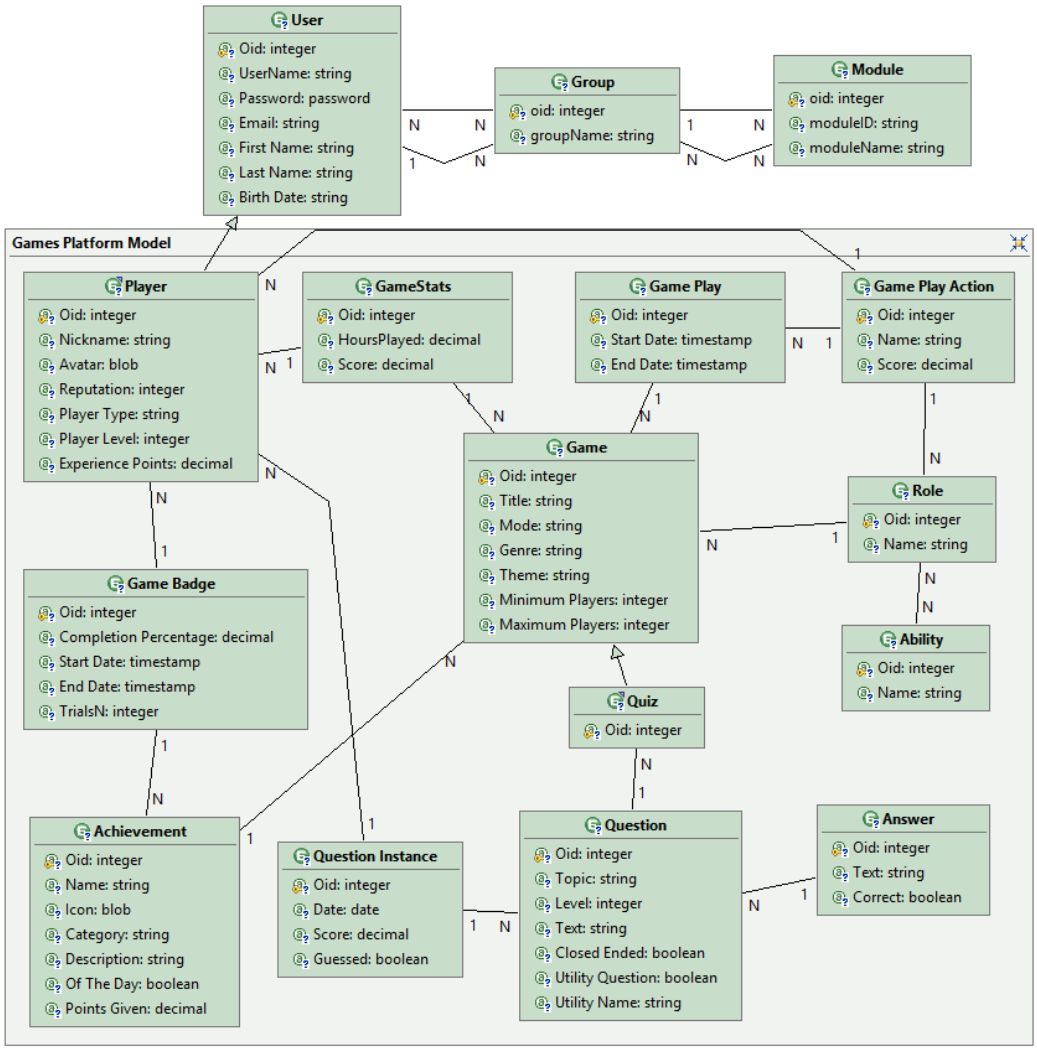


**Figure 4. Game platform data model**

# 4. Database prototype

The SmartH2O database is implemented in MySQL 5.6.14. A replica of the database is dedicated to each water utility involved in the project.

This section provides an overview of different services deployed within the platform.

## 4.1 Examples of endpoint and access procedure description

Customers interested in the SmartH2O db will access it by using two Web portals, one developed for TWUL customers and the other for SES customers, which provide secure and controlled access to remote users, allowing multiple devices (laptops, tablets, and smartphones) access to SmartH2O information.
They will consist in a read-only REST endpoint to the SmartH2O gamification portal and dataset. The endpoint will provide read-only access to the SmartH2O datasets, allowing the customers to know their total consumptions, the consumptions for different devices, etc..

The gamification engine supports the integration with the SmartH2O db by means of RESTful web services with the response available in JSON format.

Each service consists of a single REST endpoint, which contains a single method for accessing a specific gamified application.

In order to **get available actions** for a specific gamified application the Get Action web service is available. The service requires the name of the gamified application as a mandatory parameter.

The endpoint is:

```
{webappUrl}/UserActivityCreditWebServiceREST/GetActions/getActions.do
```

A sample of JSON response is:

```
{"actions": [
    {
    "gamifiedApplication": "Energy Portal",
    "actionName": "Do energy saver quiz",
    "actionID": 4
  },
    {
    "gamifiedApplication": "Energy Portal",
    "actionName": "Login",
    "actionID": 1
  }
]}
```

In order to **get the user credits** obtained by interacting with the game, it is required to specify the user email as parameter of the REST web service.

The endpoint is as follows:

```
{webappUrl}/UserActivityCreditWebServiceREST/GetUserCredits/getUserCredits.do?userEmail=
xxx@yyy.com
```

and the JSON response can be as follows:

```
{"userCredits": {
  "userEmail": "xxx @yyy.com",
  "totalCredit": 3400,
  "creditsSpent": 0,
  "creditsAvailable": 3400
```

```
}}
```

To **get the rewards** that can be redeemed by the user, it is also required to specify the user email as a parameter of the web service.

The endpoint is as follows:

```
{webappUrl}/UserActivityCreditWebServiceREST/GetUserRewards/getUserRewards.do?userEmail=
xxx@yyy.com
```

and the JSON response:

```
{"rewards": [
    {
    "rewardName": "Coupon Discount 20%",
    "rewardID": 1,
    "neededPoints": 1000,
    "userEmail": " xxx@yyy.com "
  }
]}
```

In order to register the user action in the gamification platform, the ***Assign Actions To User*** web service is available. The request is a JSON array with the following parameters:

- **email**: the email of the user to assign the action [MANDATORY]
- **time**: the timestamp of the request in Unix Timestamp format [MANDATORY]
- **area**: the name of the gamified application [MANDATORY]
- **name**: the name of the action [MANDATORY]
- **description**: the description of the action [MANDATORY]
- **tag**: additional parameter for managing non-repeatable action [NOT MANDATORY]
- **link:** additional parameter for managing non-repeatable action [NOT MANDATORY]
- **executor:** additional parameter for managing non-repeatable action [NOT MANDATORY]
- **objectkey:** additional parameter for managing non-repeatable action [NOT MANDATORY]

A sample JSON array for the request is the following:

```
[{"email":"xxx@yyy.com","time":1407307785347,"area":"Energy
Portal","name":"Login","description":"Login","tag":" ","link":" ","executor":" "}]
```

URL:

```
{webappUrl}/UserActivityCreditWebServiceREST/AssignActionsToUsers/assignActionsToUsers.do
```

In order to register the user reward in the gamification platform the ***Reedem User Reward*** we service is available. The request is a JSON array with the following parameters:

- **idReward**: the id of the reward to reedem [MANDATORY]
- **userEmail**: the email of the user that reedems the reward [MANDATORY]

An example of JSON array for the request:

```
{"idReward":1,"userEmail":" xxx@yyy.com "}
```

The endpoint is as follows:

```
{webappUrl}/ UserActivityCreditWebServiceREST/RedeemUserReward/redeemUserReward.do
```

To push user registration data about a new user in the gamification the **User Registration**

web service will be used.

The request is a JSON array with the following parameters:

- **birthdate**: the birthdate of the user in UNIX timestamp format
- **username**
- **password**
- **email**
- **firstname**
- **lastname**
- **city**
- **country**
- **publicprofile:** boolean value to indicate if the user is active or not in the community
- **internal:** boolean value to indicate if the user in an internal user of the community
- **isocode:** language isocode (used to manage international community)
- **geoarea**
- **photoname:** the name of the photo of the user
- **photocode:** the photo of the user in Base64 format

An example of the JSON request is:

```
[{"birthdate":1407276000000,"username":"markross","password":"markross","email":"mark.ross@ee.com","firstname":"Mark","lastname":"Ross","city":"London ","country":"United Kingdom","publicprofile":true,"internal":false,"isocode":"en","geoarea":"Europe"}]
```

The endpoint is as follows:

```
{webappUrl}/UserRegistrationWebServiceREST/UserRegistration/userRegistration.do
```

To push the update user data to the gamification platform the ***User update*** *web service will be used.*

The JSON array for the request is composed by the following parameters:

- **birthdate**: the birthdate of the user in UNIX timestamp format
- **username**
- **password**
- **email [MANDATORY]**
- **firstname**
- **lastname**
- **city**
- **country**
- **publicprofile:** boolean value to indicate if the user is active or not in the community
- **internal:** boolean value to indicate if the user in an internal user of the community
- **isocode:** language isocode (used to manage international community)
- **geoarea**
- **photoname:** the name of the photo of the user
- **photocode:** the photo of the user in Base64 format

An example of JSON array for the request is :

```
[{"birthdate":1407276000000,"username":"markross","password":"markross","email":"mark.ross@ee.com","firstname":"Mark","lastname":"Ross","city":"London ","country":"United Kingdom","publicprofile":true,"internal":false,"isocode":"en","geoarea":"Europe"}]
```

The endpoint is:

```
{webappUrl}/UserRegistrationWebServiceREST/UserUpdate/userUpdate.do
```

## 4.2 Prototype population

The SmartH2O db has been partially populated with synthetic data provided by partners.

In particular, on one hand some tuples produced by a preliminary version of the gamification engine have been inserted into the User Gaming Model part of the db.

On the other hand, the primary data collection step, used to record the volume of water used in a set of houses located in London, was performed by using an Access database provided by TWUL.

In particular, the **MeterReading** table was populated by using data related to cumulative meter readings produced every 15 minutes. The **House** table contains tuples related to 1200 houses located in London of different types (e.g. detached, semi-detached, etc.).

## 4.3 Data acquisition model description

Data acquisition is an essential part of water-use data management, analysis, and use since information that are efficiently produced and managed can be useful for future purpose with little additional effort. In this section we describe a model and plan for the acquisition when real data will become available.

There are three different groups of water-use data:

1. Data coming from the user gaming model: they will be integrated into the db by using RESTful web services, as described in Section 4.1.
2. Data in the consumer data model not provided by sensors: in this set of data we include information for identifying houses, users, billing prices, etc. They will be provided in Access (as in the population preliminary phase) or standard format (e.g. XML, CVS) and integrated in the SmartH2O db.
3. Data in the data model produced by sensors: this set contains both weather information and data related to rate or volume of water-use. They can be produced with a certain frequency (e.g. every 15 minutes those produced by smart meters) but then transmitted in a single shot once a day once by using JSON files. Such files will be processed in parallel by Apache Hadoop and Apache Pig scripts. Otherwise, the SmartH2O model can provide API to push such information in the db.

# 5. Data Governance Policy

The SmartH2O project develops a platform with which humans will interact, possibly exchanging information which is private and sensitive.

The SmartH2O project has declared the general principles guiding its management of ethical issues in the project proposal. We report the project stance on ethical issues in the next two sections.

## 5.1 Ethical issues related to privacy

Each party shall be responsible for ensuring its own compliance with all laws and regulations applicable to its activities, including without limitation the acquisition of data, the processing of data by it through any tool used in connection with the Project and the use of such date within the project framework. Such laws include, but are not limited to, those in respect of rights of privacy, publicity, reputation and intellectual property rights, including patent and copyright rights.

Each party shall be solely responsible for the selection of specific database vendors/data collectors/data providers, and for the performance (including any breach) of its contracts between it and such database vendors/data collectors, to which no other project partner shall be a party, and under which no other Contributor assumes any obligation or liability and shall further warrant that it has the authority to disclose the information, if any, which it provides to the other parties, and that where legally required and relevant, it has obtained appropriate informed consents from all the individuals involved.

Any party which provides any recorded data or information to another party in connection with the project will not include any information as defined by Article 2 section (a) of the European Data Protection Directive, i.e. any information relating to an identified or identifiable natural person or data subject, where an 'identifiable person' is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his or her physical, physiological, mental, economic, cultural or social identity ("Personal Data").

To this end, the providing party will take all necessary steps to ensure that the Information is "de-identified", i.e. that all Personal Data is removed from the provided information, made illegible, or otherwise made inaccessible to the receiving parties prior to provision.

## 5.2 Ethical issues related to the involvement of users

Provided that the privacy of users' data will be adequately covered, as described above, all users involved will be explicitly requested for their consent regarding their participation in the study. The data privacy policy will be clearly communicated and it will be possible to withdraw from the project at any stage. A withdrawal will imply the complete and permanent removal of all the users' data from the project database.

In the case of the involvement of children in activities promoted in collaboration with the primary and secondary schools in the Swiss case study, the consent to participate in the project will be requested in written form, to be signed by the children's parents or legal tutors.

The data collected by the SmartH2O project can be of sensitive nature, as it contains detailed information about household water consumption correlated with socio-economic and psychographic characteristics, which, if abused, misused and processed without the data owner consent, could bring severe damage to both the individual consumers and the reputation of the water utilities, and the SmartH2O project.

## 5.3 The SmartH2O data governance policy

In order to manage personal data in accordance to the above defined principles, a strict data governance policy will be thus enforced. It is our opinion that an inherent quality of a data governance policy is its simplicity. If it is simple and clear, it will be easier to adhere to it, to implement, and to make sure it is properly enforced.

Our data governance policy revolves around the following principles:

- the water utility is the owner of water meter data
- the consumer is the owner of the psychographic data (values, opinions, attitudes, interests) and of socio-economic data (household type, number of occupants, level of income, etc.)
- the water utility has the right to select which subset of its water meter data connect to the SmartH2O database;
- the consumer has the right to access all his/her data, and has also the right to terminate its participation in the SmartH2O platform at any time.

Our data governance policy defines the rights for data protection:

- The consumer and the water utility have the right to have their data being adequately protected from violations. Data must be secured and only authorised partners of the SmartH2O project can access it.
- The water utility data will be accessible only to those partner institutions who have signed a non-disclosure agreement with the water utility. The water utility can renounce to such an agreement, and grant access to all partners in the project.
- The consumer data will be made accessible to all SmartH2O project partners only with the user consent. The user gives his/her consent by accepting a Terms of Use agreement when signing in for the first time on the SmartH2O platform.

Our data governance policy specifies the technological solution adopted to guarantee that the principles and the data protection rights are enforced:

- Only the user will have access to data regarding his/her identity: home address, name and surname are separated from all other data.
- User specific data are transmitted over a secured connection to the SmartH2O platform, where they are safely protected. Such data will not leave the SmartH2O platform, in order to minimise the risk of interception and to have a single failure point.
- All SmartH2O user profiling algorithms will thus process anonymised data. Anonymised data will be transmitted by the SmartH2O platform to the SmartH2O partners' servers in order to process them.

The data governance policy defines the technical solutions to protect the data during the work flows of the development phase. It regards:

a. Data in transit (data that is transferred between the SmartH2O development server and other network nodes)
    - Firewall FortiGate 300C. No external non-authorized access;
    - IPSEC VPN access for listed IPs. The partners will send their authorized IPs. They will receive a ready configured VPN client. The partners will connect through VPN in order to access the platform applications and services;
    - Secured FTP to upload data files to the development SmartH2O server. The FTP server uses a custom configuration;
    - Secured HTTP for application access over the internet;
    - User / password authetication for application access over the internet;

b. Data in use (active data under constant change, stored in RAM)
    - Processing data on unique physical server with unique external IP;
    - Setting access rights for data manipulation at application level;
    - Implementing encryption protocols for accesing data via web-services;
    - Application whitelistening;

c. Data at rest (inactive data stored in off-site database backups, archives, tapes, CDs/DVDs, USB sticks)
    - Private datacenter with limited access (4 IT staff) card based;
    - Unique IT admin for the development SmartH2O server;
    - Policy of No Off-site data backup allowed;

# 6. Data management tools

SmartH20 uses data amangement tools at both runtime and design time.

- Runtime data management tools comprise the Smart Meter Data Management Component (SMDMC), which has been built ad hoc to support the acquisition of consumption data from heterogeneous smart sensor infrastructures.
- Design time data management tool address the modelling and creation of the database schema code, which supports the creation and evolutive maintenance of the SQL schema of the SmartH20 database. This task has been addressed using the Domain Modeller of the WebRatio tool suite.

## 6.1 Smart Meter Data Management Component – SMDMC

The Smart Meter Data Management component implements the data acquisition and data assimilation in the SmartH2O Database.

### *6.1.1 Role and Functionality*

This platform component implements Use Case 8.1 stated in D2.2 Final Requirements deliverable: “Collecting consumption data with smart meters”. This component was designed to acquire and consolidate raw water metered usage consumption data. Its main functional and technical requirements are presented below:

- Facilitate Water Utility company the required communication infrastructure to transfer water metered usage data to SmartH2O platform
- Process received data and save results into SmartH2O platform database
- Ensure logging of data processing
- Ensure data security and integrity during transfer, processing and storage stages
- Ensure a scalable computing and storage architecture able to process large amounts of data sent with high-frequency

A more detailed description of component’s functional requirements will be found in D2.3 Functional Requirements Deliverable (not yet delivered).

### *6.1.2 System Flow*

The structure of the network components where the various elements of the SMDMC architecture are deployed is presented in Figure 5.

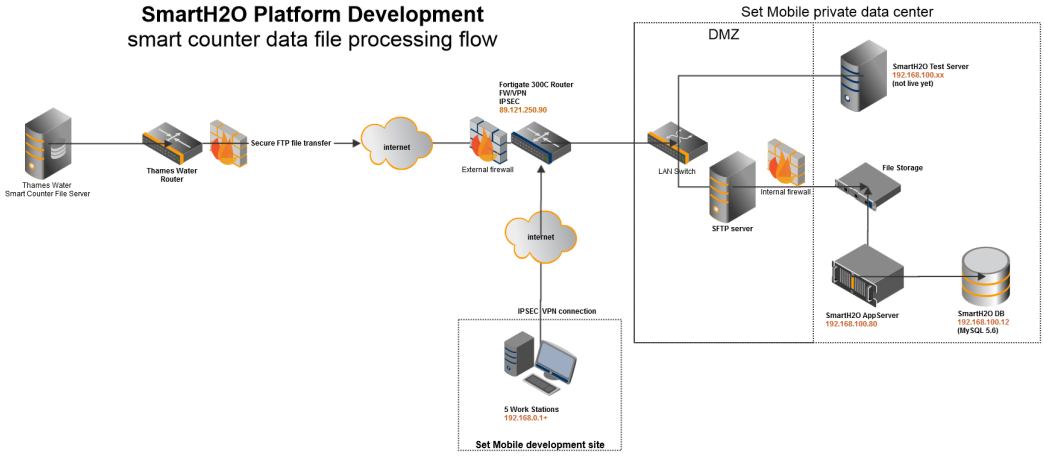

**Figure 5. A graphical overview of the network architecture of SMDMC.**

The high level workflow of transferring, receiving, storing and processing of the smart counter files is made of the following steps:

**Step 1**: the Smart H2O Admin (the Receiver of data) transmits to Water Utility (the Sender of data) the parameters and credentials for accessing the Secured FTP (SFTP) server. The SFTP server resides in the DMZ of a data center protected firewall router.

**Step 2**: The Provider connects, authenticates and uploads the data files containing smart water counter readings to the SFTP server. After successfully receiving the files, the Sender then moves the data files to the File Storage, which resides in the non-DMZ LAN of the data center.

**Step 3**: The providing partner will upload the MD5 signature of the uploaded archive. This will be used by the Receiver for successfully validating the file transfer.

**Step 4**: The SMDMC running on the SmartH2O application server process the data files and stores the data in a local database protected by a build-in security layer. The processing consists in:

- parsing received files,
- detecting and report data inconsistencies,
- aggregating counter consumption to household level,
- saving aggregated data in database.

**Step 5**: After processing, the data files are automatically encrypted and moved in a dedicated zone on File Storage. A log will be available for the Sender partner to acknowledge the outcome of the process.

**Step 6**: The data saved in the database is accessed, processed and displayed by user/password authenticated applications according to the business logic.

### *6.1.3 Architecture and Deployment*

To ensure processing scalability, BigData technologies were employed in design of SMDMC component. The data files processing technology is based on Apache Hadoop components and infrastructure for distributed and parallel processing. The deployment architecture of SMDMC is presented in Figure 6.

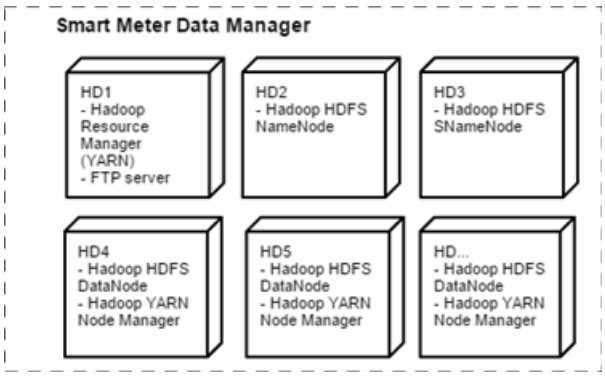


**Figure 6. The deployed components of SMDMC.**

Due to its distributed architecture, the processing power can be easily increased by increasing the number of processing nodes of Apache Hadoop, similar to node HD4 and HD5.

The Development Environment deployment is based on WMWare ESXi virtualization solution.

The Production Environment deployment can be done on OpenStack virtualization layer that can manage large number of hardware resource providing scalability and high-availability.

A more detailed description of the component's architecture and deployment can be found in D6.2 Platform Architecture and Design

### *6.1.4 Data Security*

We have adopted technical solutions to protect the water usage data files during the transfer, processing and storage phases. In synthesis these are:

d. Data in transit (data that is transferred between the SmartH2O development server and other network nodes):
    - Router Firewall. No external non-authorized access;
    - IPSEC VPN access for listed IPs. The partners will send their authorized IP addresses. They will receive a ready configured VPN client. The partners will connect through VPN in order to access the platform applications and services;
    - *Secured FTP* to upload data files to the development SmartH2O server. The FTP server uses a custom configuration;
    - Secured HTTP for application access over the internet;
    - User / password authentication for application access over the internet;

e. Data in use (active data under constant change, stored in RAM)

    - Processing data on unique physical server with unique external IP;
    - Setting access rights for data manipulation at application level;
    - Implementing encryption protocols for accessing data via web-services;
    - Application *whitelistening*;

f. Data at rest (inactive data stored in off-site database backups, archives, tapes, CDs/DVDs, USB sticks)
    - Private datacenter with limited access (4 IT staff) card based;
    - Unique IT Admin for the development SmartH2O server;
    - Policy of No Off-site data backup allowed;

## 6.2 WebRatio Domain Modeler

***WebRatio***[11] is a tool that supports database, SOA, BPM, web and mobile application design, exploiting a conceptual modelling approach coupled to code generation. WebRatio covers the development phases of data design and application design, and supports implementation by automating the production of the relational database and of the application interfaces. More precisely, WebRatio focuses on five main aspects:

- *Data design:* it supports the design of Entity-Relationship data schemas, with a graphical user interface for drawing and specifying the properties of entities, relationships, attributes, and generalization hierarchies.
- *Application design:* it assists the design of interfaces for web and mobile applications, providing functions for drawing and specifying the properties of such as artefacts view containers, areas, pages, components, and interaction flows, expressed using the OMG IFML standard[12].
- *Data Mapping:* it permits declaring the set of data sources to which the conceptual data schema has to be mapped, and automatically translates Entity-Relationship

[11] www.webratio.com

[12] www.omg.org/spec/ifml

diagrams and UML OCL expressions into relational database tables and views.

- *Presentation design:* it offers functionality for defining the presentation style of the application, allowing the designer to create style sheets and associate them to interface elements, and organize page layout, by arranging the relative position of components in the page.
- *Code generation:* it automatically translates conceptual models into running Web applications built on top of the JEE architecture.

### *6.2.1 Database design*

WebRatio provides a graphical user interface, which allows designers to compose the Entity-Relationship diagram corresponding to the database that will host the data of the application.

Figure 7 shows a snapshot of the WebRatio user interface, which is organized into the typical four areas of application development tools:

- A project tree (upper left frame), organising all the elements of the application project.
- A work area (upper right frame), where the specifications are visually edited.
- A property frame (lower left frame), where the properties of individual elements can be set.
- A message area (lower right frame), where messages and warnings are displayed.

In particular, Figure 7 shows the Entity-Relationship diagram of a sample database design. The work area visualizes the data schema, and the designer can define entities, attributes, relationships, and generalizations.

The elements displayed in the diagram are also presented in the project tree, where they are hierarchically organized in folders. The properties of the currently selected element of the schema are displayed and can be edited in a property frame.

A WebRatio application project consists of one or more Entity-Relationship diagrams and of a set of IFML site view specifications.

WebRatio gives support to the design of the database support for applications using a Role Based Access Control (RBAC) approach.

To this end, a default database conceptual schema consisting of the User, Group, and Module entities and of their RBAC relationships is automatically added to each project, and the developer can extend it with additional entities and relationships reflecting further aspects of the permission system he wants to design.

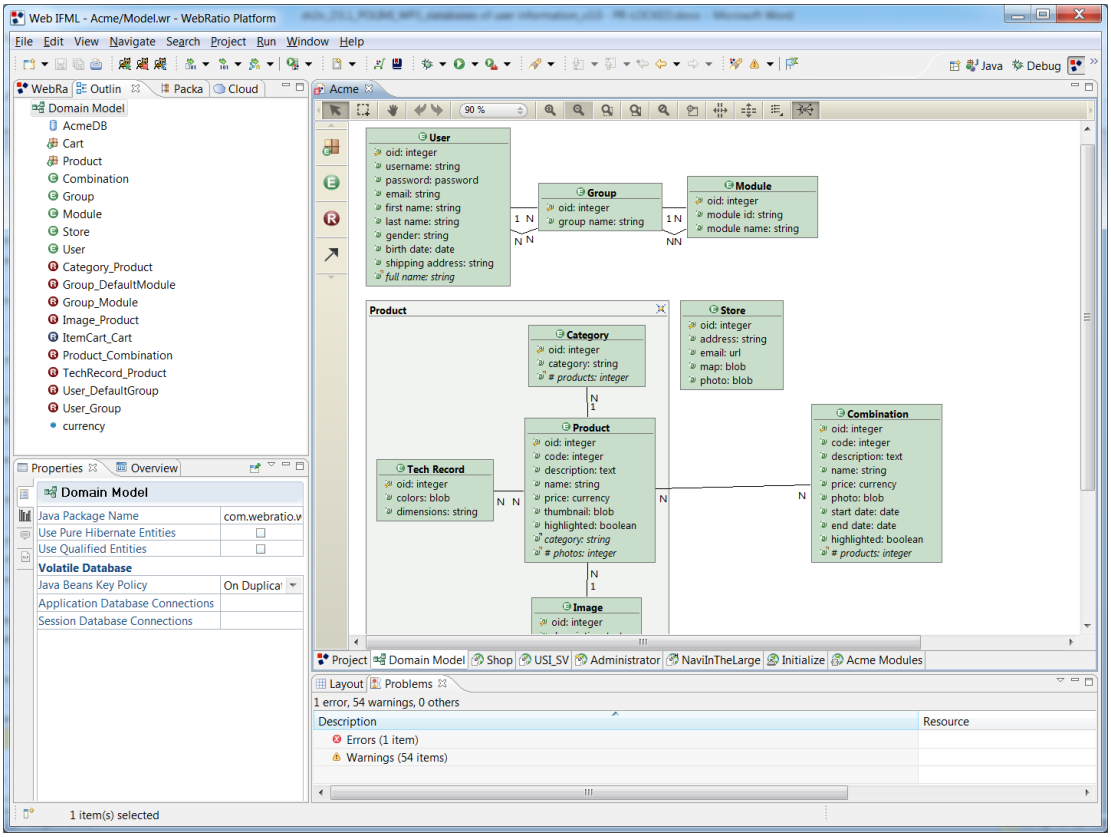


**Figure 7: Database design in WebRatio**

WebRatio supports also the visual definition of derived data, i.e., data that is calculated based on other stored or calculated data.

A wizard (Figure 8) can be invoked to specify the expression for computing a derived entity, attribute or relationship.

Such expression, written in a subset of the OCL language, is automatically translated into a SQL view, installable into the application database.

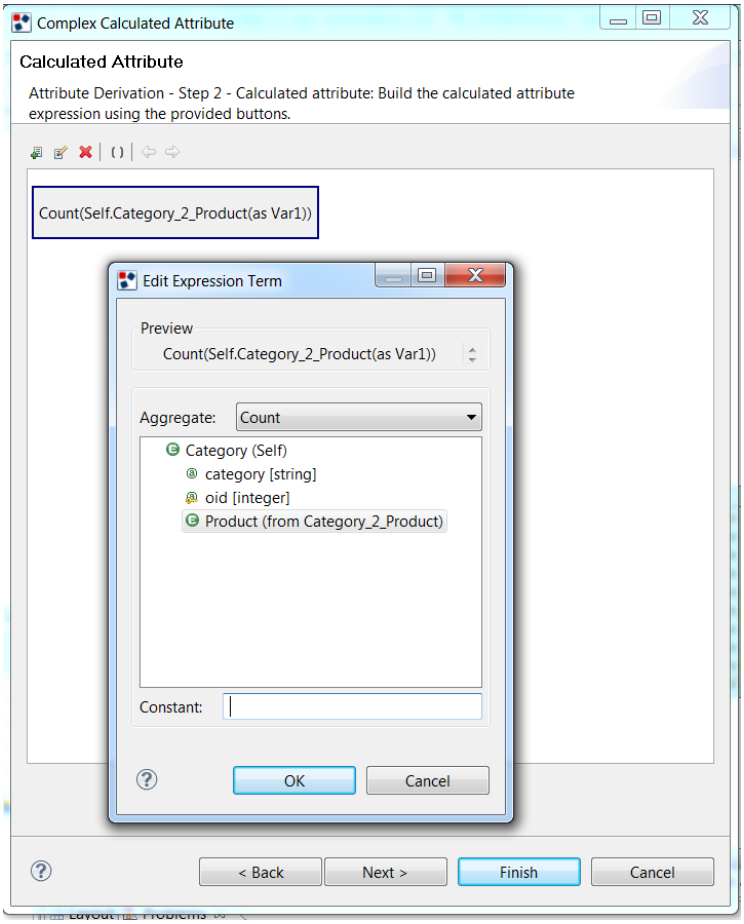


**Figure 8: Derivation wizard, at work on the definition of a calculated attributed**

### *6.2.2 Data mapping and database creation*

WebRatio assists the data implementation phase, by associating the application to the data sources where content resides. Three data implementation architectures described in are supported (dedicated, replicated, and online database), with the highest level of assistance for the dedicated database solution.

- **Dedicated Database**: this situation occurs when the content *does not exist prior to the development of the application*. In this case, the development of the Web or mobile application comprises also the construction of a dedicated database, purposely built for storing the content to be published. Content maintenance is done with an ad hoc application, for example, with a content management interface. Typical applications with dedicated databases are B2C and corporate portals, which are conceived specifically to collect and deliver content that is not reused outside the Web / mobile application.
- **Replicated Database**: this situation occurs when the content is stored in one or more corporate data sources, for example in an operational databases or legacy systems, and is periodically copied into a database dedicated to the Web application. The *Web or mobile application owns and publishes a read-only copy of the corporate data* and the original content continues to be created and updated in its native location. An example of this scenario could is the SmartH20 consumer portal that publishes consumption data maintained in the utility data collection system.
- **On-line Database:** *the Web application has direct access to the corporate data*, to publish the current version of the content. In this case, the Web application has no

dedicated database but connects directly to the external data sources, for either reading or writing content. An example of this category of applications is a Web-based or mobile reservation system, allowing users to see and change the up-to-date version of the reservation database.

The connection to the data sources exploits the JDBC and Hibernate APIs; additional kinds of data sources can be added, by programming the services for connecting to them.

The data implementation activity proceeds by mapping the Entity-Relationship diagram onto the defined data sources; the user declares the data sources, and binds entities and relationships to tables. The mapping information, associating entities, relationships, and attributes with tables and columns, is stored in a configuration file.

If the database for the application content does not exist, WebRatio can automatically create the default standard database, by applying the standard translation rules from entity and relationships to relational tables, with a Generate SQL command.

The tool automatically creates the standard tables and binds the entities and relationships of the project to them. Then the user populates the database manually or with a data replication tool. Figure 9 shows the SQL code generated by WebRatio from the ER schema of Figure 7.

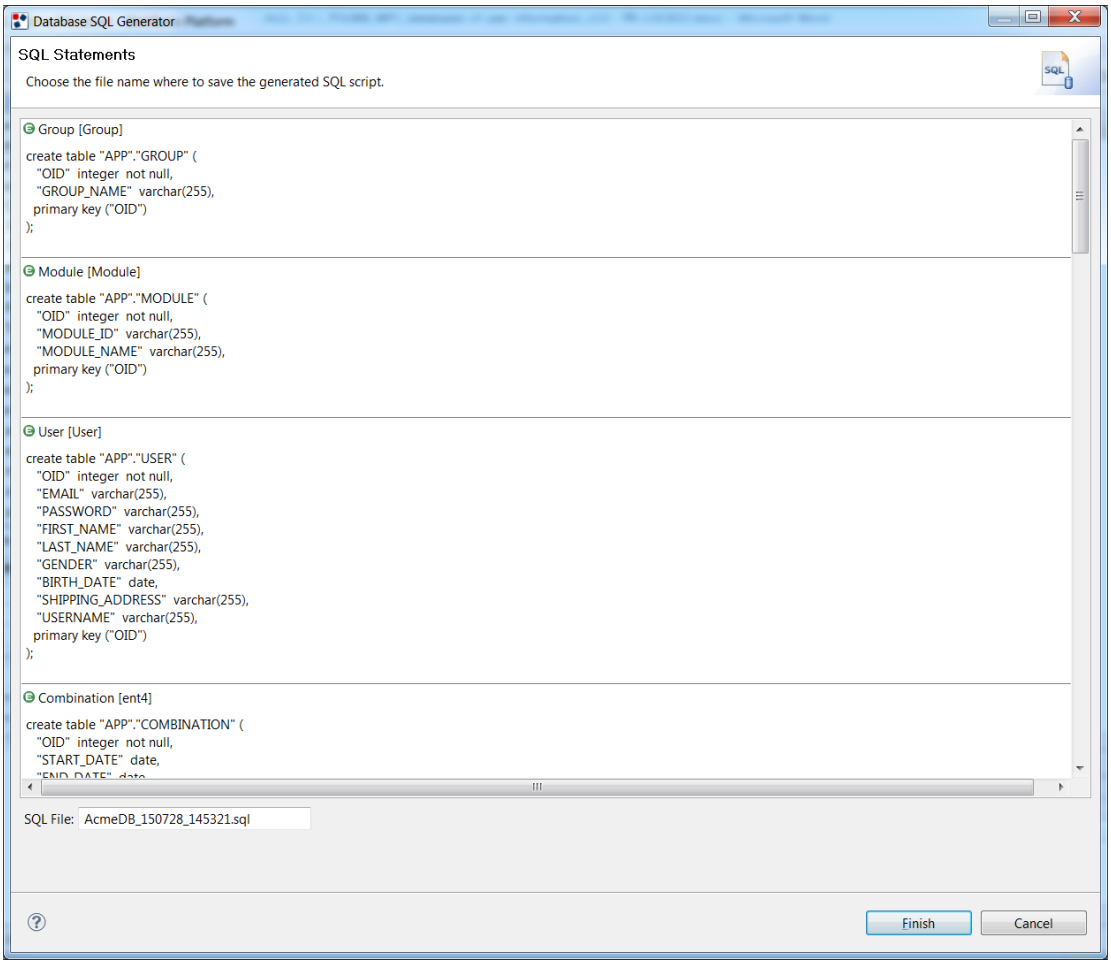


**Figure 9: The output of the Generate SWL command**

If Entity-Relationship schema contains derived data, the generation command translates the OCL expressions of the derived schema elements, and produces a source file containing the SQL statements defining the relational views equivalent to the OCL expressions, which can be automatically or manually installed into the appropriate data source.

All entities, relationships, and derived elements must be correctly mapped before generating the code and running the application, otherwise the code generation may produce incomplete results. WebRatio also support database evolution and maintenance.

The tool can check the alignment between the Entity-Relationship diagram and the physical databases, thus facilitating the tracking of changes in either of the two levels. Figure 10 shows the interface of the Database Synchronizer wizard, which helps the developer assess the changes in the ER diagram and in the database relational schema that need to be propagated after a change.

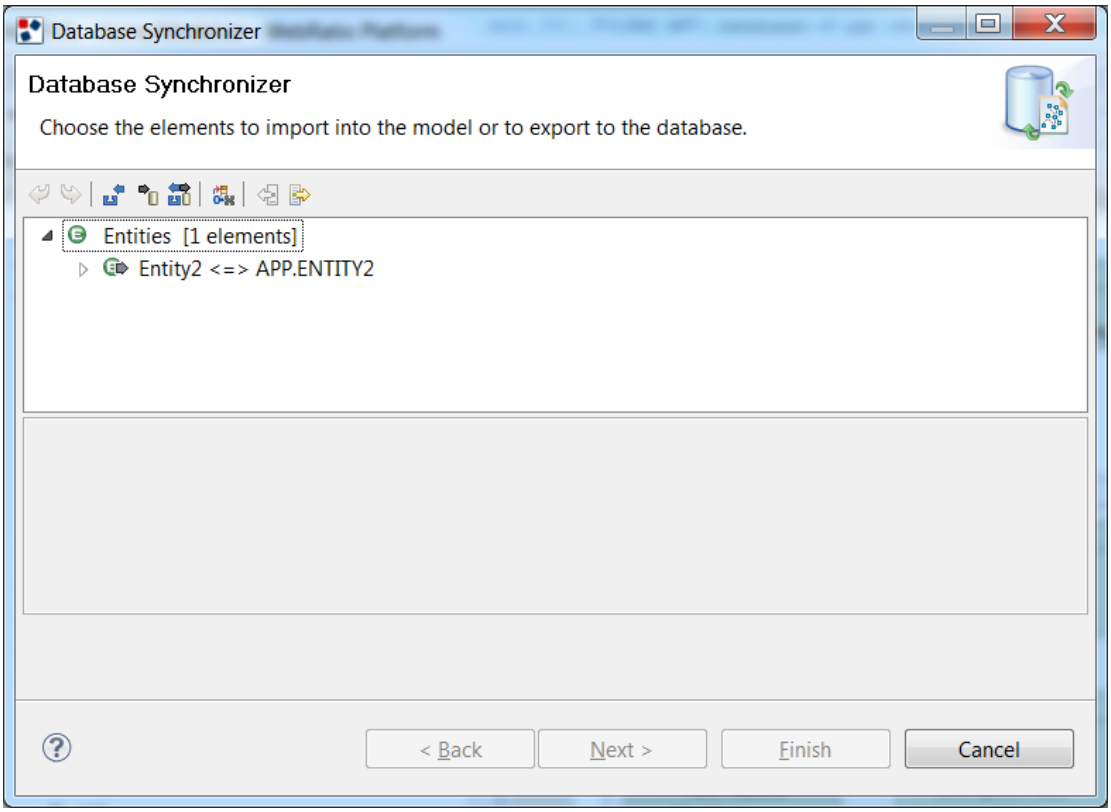


**Figure 10: Two-way ER-to-Database synchronization command**

# 7. Conclusions and future work

The conducted review on past residential water end use studies, along with initial iterations with water consumers and with the water utilities taking part at the SmartH2O project (i.e., TWUL and SES), has led to the identification of a set of potentially relevant variables influencing water consumption. Such variables are included in the SmartH2O database, which has been defined in terms of Entity-Relationship models. A first prototype of the SmartH2O database has been implemented in MySQL 5.6.14. The database structure has been deliberately kept open and flexible to accommodate additional information coming from further interactions with the water utilities and the end users.

The database will be populated with data and information on the users provided by the water utilities or by the users themselves (e.g., through meter readings, surveys and the *game with a purpose* application developed in WP4). Automated procedures to populate the database have been proposed and they will be implemented in SmartH2O platform as soon as data will become available.

The data that will be gathered will be used for disaggregating water flow data into different water end use categories and for profiling water users through machine learning and data-mining algorithms that will be developed in Task 3.2 of the project.

# 8. Appendix Database creation SQL code

## 8.1 Consumer Portal subschema

```
-- MySQL dump 10.13  Distrib 5.6.17, for Win32 (x86)
--
-- Host: localhost    Database: consumer_portal_db_v4
-- ------------------------------------------------------
-- Server version 5.6.23-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014      SET      @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101                SET                @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `alert`
--

DROP TABLE IF EXISTS `alert`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `alert` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `date` datetime DEFAULT NULL,
  `neutral_user_oid` int(11) DEFAULT NULL,
  `mail_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_alert_neutral_user` (`neutral_user_oid`),
  KEY `fk_alert_mail` (`mail_oid`),
  CONSTRAINT  `fk_alert_mail`  FOREIGN  KEY  (`mail_oid`)  REFERENCES
`mail` (`oid`),
  CONSTRAINT        `fk_alert_neutral_user`        FOREIGN        KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `bill`
--

DROP TABLE IF EXISTS `bill`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `bill` (
  `oid` int(11) NOT NULL,
  `account_number` varchar(255) DEFAULT NULL,
  `bill_date` date DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `volume_charge` decimal(19,2) DEFAULT NULL,
  `service_charge` decimal(19,2) DEFAULT NULL,
  `currency` varchar(255) DEFAULT NULL,
  `volume_eur_charge` decimal(19,2) DEFAULT NULL,
  `service_eur_charge` decimal(19,2) DEFAULT NULL,
  `exchange_rate` decimal(19,2) DEFAULT NULL,
  `exchange_date` date DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_bill_household` (`household_oid`),
  CONSTRAINT `fk_bill_household` FOREIGN KEY (`household_oid`)
REFERENCES `household` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `billing_price`
--

DROP TABLE IF EXISTS `billing_price`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `billing_price` (
  `oid` int(11) NOT NULL,
  `month` varchar(255) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `monthly_service_charge` decimal(19,2) DEFAULT NULL,
  `monthly_volume_charge` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `billing_price_bill`
--

DROP TABLE IF EXISTS `billing_price_bill`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `billing_price_bill` (
  `billing_price_oid` int(11) NOT NULL,
  `bill_oid` int(11) NOT NULL,
  PRIMARY KEY (`billing_price_oid`,`bill_oid`),
  KEY `fk_billing_price_bill_billing` (`billing_price_oid`),
  KEY `fk_billing_price_bill_bill` (`bill_oid`),
  CONSTRAINT `fk_billing_price_bill_bill` FOREIGN KEY (`bill_oid`)
REFERENCES `bill` (`oid`),
  CONSTRAINT `fk_billing_price_bill_billing` FOREIGN KEY
(`billing_price_oid`) REFERENCES `billing_price` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `building`
--

DROP TABLE IF EXISTS `building`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `building` (
  `oid` int(11) NOT NULL,
  `building_garden_area` decimal(19,2) DEFAULT NULL,
  `building_pool_volume` decimal(19,2) DEFAULT NULL,
  `age` int(11) DEFAULT NULL,
  `building_size` decimal(19,2) DEFAULT NULL,
  `residence_type` varchar(255) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  `building_garden` bit(1) DEFAULT NULL,
  `building_pool` bit(1) DEFAULT NULL,
  `district_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_building_district` (`district_oid`),
  CONSTRAINT `fk_building_district` FOREIGN KEY (`district_oid`)
REFERENCES `district` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `consumer_segment`
--

DROP TABLE IF EXISTS `consumer_segment`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `consumer_segment` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `consumer_segment_neutral_user`
--

DROP TABLE IF EXISTS `consumer_segment_neutral_user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `consumer_segment_neutral_user` (
  `consumer_segment_oid` int(11) NOT NULL,
  `neutral_user_oid` int(11) NOT NULL,
  PRIMARY KEY (`consumer_segment_oid`,`neutral_user_oid`),
  KEY `fk_consumer_segment_neutral_us` (`consumer_segment_oid`),
  KEY `fk_consumer_segment_neutral_2` (`neutral_user_oid`),
  CONSTRAINT     `fk_consumer_segment_neutral_2`     FOREIGN     KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
  CONSTRAINT     `fk_consumer_segment_neutral_us`     FOREIGN     KEY
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `device_class`
--

DROP TABLE IF EXISTS `device_class`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `device_class` (
```

```
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `number` int(11) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_device_class_household` (`household_oid`),
  CONSTRAINT `fk_device_class_household` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `device_consumption`
--

DROP TABLE IF EXISTS `device_consumption`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `device_consumption` (
  `oid` int(11) NOT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `device_consumption` decimal(19,2) DEFAULT NULL,
  `device_class_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_device_consumption_device_c` (`device_class_oid`),
  CONSTRAINT `fk_device_consumption_device_c` FOREIGN KEY
(`device_class_oid`) REFERENCES `device_class` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `district`
--

DROP TABLE IF EXISTS `district`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `district` (
  `oid` int(11) NOT NULL,
  `zipcode` varchar(255) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `feature`
--

DROP TABLE IF EXISTS `feature`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `feature` (
  `oid` int(11) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `consumer_segment_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_feature_consumer_segment` (`consumer_segment_oid`),
  CONSTRAINT      `fk_feature_consumer_segment`      FOREIGN      KEY
(`consumer_segment_oid`) REFERENCES `consumer_segment` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `group`
--

DROP TABLE IF EXISTS `group`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `group` (
  `oid` int(11) NOT NULL,
  `groupname` varchar(255) DEFAULT NULL,
  `module_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_group_module` (`module_oid`),
  CONSTRAINT `fk_group_module` FOREIGN KEY (`module_oid`) REFERENCES
`module` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `group_module`
--

DROP TABLE IF EXISTS `group_module`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `group_module` (
  `group_oid` int(11) NOT NULL,
  `module_oid` int(11) NOT NULL,
  PRIMARY KEY (`group_oid`,`module_oid`),
  KEY `fk_group_module_group` (`group_oid`),
  KEY `fk_group_module_module` (`module_oid`),
  CONSTRAINT `fk_group_module_group` FOREIGN KEY (`group_oid`)
REFERENCES `group` (`oid`),
  CONSTRAINT `fk_group_module_module` FOREIGN KEY (`module_oid`)
REFERENCES `module` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `household`
--

DROP TABLE IF EXISTS `household`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `household` (
  `oid` int(11) NOT NULL,
  `utilityid` varchar(255) DEFAULT NULL,
  `household_size` decimal(19,2) DEFAULT NULL,
  `ownership` bit(1) DEFAULT NULL,
  `number_occupants` int(11) DEFAULT NULL,
  `number_pets` int(11) DEFAULT NULL,
  `household_garden_area` decimal(19,2) DEFAULT NULL,
  `household_pool_volume` decimal(19,2) DEFAULT NULL,
  `second` bit(1) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `visible` bit(1) DEFAULT NULL,
  `pets` bit(1) DEFAULT NULL,
  `household_pool` bit(1) DEFAULT NULL,
  `household_garden` bit(1) DEFAULT NULL,
  `family_household` bit(1) DEFAULT NULL,
  `consumption_range` varchar(255) DEFAULT NULL,
  `saving_motivation` varchar(255) DEFAULT NULL,
  `family_id` varchar(255) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_smart_meter` (`smart_meter_oid`),
```

```
  KEY `fk_household_building` (`building_oid`),
  CONSTRAINT `fk_household_building` FOREIGN KEY (`building_oid`)
REFERENCES `building` (`oid`),
  CONSTRAINT `fk_household_smart_meter` FOREIGN KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `household_consumption`
--

DROP TABLE IF EXISTS `household_consumption`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `household_consumption` (
  `oid` int(11) NOT NULL,
  `consumption` decimal(19,2) DEFAULT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_household_consumption_house` (`household_oid`),
  CONSTRAINT `fk_household_consumption_house` FOREIGN KEY
(`household_oid`) REFERENCES `household` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `mail`
--

DROP TABLE IF EXISTS `mail`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `mail` (
  `oid` int(11) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `body` longtext,
  `language` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
-- Table structure for table `media_asset`
--

DROP TABLE IF EXISTS `media_asset`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `media_asset` (
  `oid` int(11) NOT NULL,
  `title` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `duration` decimal(19,2) DEFAULT NULL,
  `author` varchar(255) DEFAULT NULL,
  `media` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `meter_reading`
--

DROP TABLE IF EXISTS `meter_reading`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `meter_reading` (
  `oid` int(11) NOT NULL,
  `reading_date_time` datetime DEFAULT NULL,
  `company` varchar(255) DEFAULT NULL,
  `total_consumption` decimal(19,2) DEFAULT NULL,
  `smart_meter_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_meter_reading_smart_meter` (`smart_meter_oid`),
  CONSTRAINT     `fk_meter_reading_smart_meter`     FOREIGN     KEY
(`smart_meter_oid`) REFERENCES `smart_meter` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `module`
--

DROP TABLE IF EXISTS `module`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `module` (
  `oid` int(11) NOT NULL,
  `moduleid` varchar(255) DEFAULT NULL,
  `moduledomainname` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `neutral_user`
--

DROP TABLE IF EXISTS `neutral_user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `neutral_user` (
  `user_oid` int(11) NOT NULL,
  `registration_date` date DEFAULT NULL,
  `family_role` varchar(255) DEFAULT NULL,
  `house_holder` bit(1) DEFAULT NULL,
  `educational_level` varchar(255) DEFAULT NULL,
  `income_rate` varchar(255) DEFAULT NULL,
  `currency` varchar(255) DEFAULT NULL,
  `public` bit(1) DEFAULT NULL,
  `language` varchar(255) DEFAULT NULL,
  `temperature_unit` varchar(255) DEFAULT NULL,
  `length_unit` varchar(255) DEFAULT NULL,
  `household_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_oid`),
  KEY `fk_neutral_user_household` (`household_oid`),
  KEY `fk_neutral_user_user` (`user_oid`),
  CONSTRAINT       `fk_neutral_user_household`       FOREIGN       KEY
(`household_oid`) REFERENCES `household` (`oid`),
  CONSTRAINT   `fk_neutral_user_user`   FOREIGN   KEY   (`user_oid`)
REFERENCES `user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `neutral_user_mediaasset`
--

DROP TABLE IF EXISTS `neutral_user_mediaasset`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `neutral_user_mediaasset` (
  `neutral_user_oid` int(11) NOT NULL,
  `media_asset_oid` int(11) NOT NULL,
  PRIMARY KEY (`neutral_user_oid`,`media_asset_oid`),
  KEY `fk_neutral_user_mediaasset_neu` (`neutral_user_oid`),
  KEY `fk_neutral_user_mediaasset_med` (`media_asset_oid`),
  CONSTRAINT     `fk_neutral_user_mediaasset_med`     FOREIGN     KEY
(`media_asset_oid`) REFERENCES `media_asset` (`oid`),
  CONSTRAINT     `fk_neutral_user_mediaasset_neu`     FOREIGN     KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `neutral_user_tip`
--

DROP TABLE IF EXISTS `neutral_user_tip`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `neutral_user_tip` (
  `neutral_user_oid` int(11) NOT NULL,
  `tip_oid` int(11) NOT NULL,
  PRIMARY KEY (`neutral_user_oid`,`tip_oid`),
  KEY `fk_neutral_user_tip_neutral_us` (`neutral_user_oid`),
  KEY `fk_neutral_user_tip_tip` (`tip_oid`),
  CONSTRAINT     `fk_neutral_user_tip_neutral_us`     FOREIGN     KEY
(`neutral_user_oid`) REFERENCES `neutral_user` (`user_oid`),
  CONSTRAINT   `fk_neutral_user_tip_tip`   FOREIGN   KEY   (`tip_oid`)
REFERENCES `tip` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `smart_meter`
--

DROP TABLE IF EXISTS `smart_meter`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `smart_meter` (
  `oid` int(11) NOT NULL,
  `smart_meter_id` varchar(255) DEFAULT NULL,
  `building_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_smart_meter_building` (`building_oid`),
```

```
  CONSTRAINT `fk_smart_meter_building` FOREIGN KEY (`building_oid`)
REFERENCES `building` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tip`
--

DROP TABLE IF EXISTS `tip`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tip` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `header` varchar(255) DEFAULT NULL,
  `body` longtext,
  `tipdate` date DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `unit_of_measurement`
--

DROP TABLE IF EXISTS `unit_of_measurement`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `unit_of_measurement` (
  `oid` int(11) NOT NULL,
  `physical_quantity` varchar(255) DEFAULT NULL,
  `primary_unit` varchar(255) DEFAULT NULL,
  `secondary_unit` varchar(255) DEFAULT NULL,
  `conversion_coefficient` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `user`
--

DROP TABLE IF EXISTS `user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user` (
  `oid` int(11) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `birth_date` varchar(255) DEFAULT NULL,
  `internal` bit(1) DEFAULT NULL,
  `group_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_user_group` (`group_oid`),
  CONSTRAINT `fk_user_group` FOREIGN KEY (`group_oid`) REFERENCES
`group` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `user_group`
--

DROP TABLE IF EXISTS `user_group`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_group` (
  `user_oid` int(11) NOT NULL,
  `group_oid` int(11) NOT NULL,
  PRIMARY KEY (`user_oid`,`group_oid`),
  KEY `fk_user_group_user` (`user_oid`),
  KEY `fk_user_group_group` (`group_oid`),
  CONSTRAINT `fk_user_group_group` FOREIGN KEY (`group_oid`)
REFERENCES `group` (`oid`),
  CONSTRAINT `fk_user_group_user` FOREIGN KEY (`user_oid`)
REFERENCES `user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `weather_condition`
--

DROP TABLE IF EXISTS `weather_condition`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `weather_condition` (
  `oid` int(11) NOT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  `rain_fall` decimal(19,2) DEFAULT NULL,
  `average_temperature` decimal(19,2) DEFAULT NULL,
  `district_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_weather_condition_district` (`district_oid`),
  CONSTRAINT `fk_weather_condition_district` FOREIGN KEY
(`district_oid`) REFERENCES `district` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2015-05-05 14:28:42
```

## 8.2 Games platform subschema

```
-- MySQL dump 10.13  Distrib 5.6.17, for Win32 (x86)
--
-- Host: localhost    Database: games_platform_db
-- ------------------------------------------------------
-- Server version 5.6.23-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
```

```
--
-- Table structure for table `ability`
--

DROP TABLE IF EXISTS `ability`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `ability` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `achievement`
--

DROP TABLE IF EXISTS `achievement`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `achievement` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `icon` varchar(255) DEFAULT NULL,
  `category` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `of_the_day` bit(1) DEFAULT NULL,
  `points_given` decimal(19,2) DEFAULT NULL,
  `game_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_achievement_game` (`game_oid`),
  CONSTRAINT   `fk_achievement_game`   FOREIGN   KEY   (`game_oid`)
REFERENCES `game` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `answer`
--

DROP TABLE IF EXISTS `answer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `answer` (
```

```
  `oid` int(11) NOT NULL,
  `text` varchar(255) DEFAULT NULL,
  `correct` bit(1) DEFAULT NULL,
  `question_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_answer_question` (`question_oid`),
  CONSTRAINT `fk_answer_question` FOREIGN KEY (`question_oid`)
REFERENCES `question` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `game`
--

DROP TABLE IF EXISTS `game`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `game` (
  `oid` int(11) NOT NULL,
  `title` varchar(255) DEFAULT NULL,
  `mode` varchar(255) DEFAULT NULL,
  `genre` varchar(255) DEFAULT NULL,
  `theme` varchar(255) DEFAULT NULL,
  `minimum_players` int(11) DEFAULT NULL,
  `maximum_players` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `game_badge`
--

DROP TABLE IF EXISTS `game_badge`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `game_badge` (
  `oid` int(11) NOT NULL,
  `completion_percentage` decimal(19,2) DEFAULT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `trialsn` int(11) DEFAULT NULL,
  `achievement_oid` int(11) DEFAULT NULL,
  `player_oid` int(11) DEFAULT NULL,
```

```
  PRIMARY KEY (`oid`),
  KEY `fk_game_badge_achievement` (`achievement_oid`),
  KEY `fk_game_badge_player` (`player_oid`),
  CONSTRAINT `fk_game_badge_achievement` FOREIGN KEY
(`achievement_oid`) REFERENCES `achievement` (`oid`),
  CONSTRAINT `fk_game_badge_player` FOREIGN KEY (`player_oid`)
REFERENCES `player` (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `game_play`
--

DROP TABLE IF EXISTS `game_play`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `game_play` (
  `oid` int(11) NOT NULL,
  `start_date` datetime DEFAULT NULL,
  `end_date` datetime DEFAULT NULL,
  `game_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_game_play_game` (`game_oid`),
  CONSTRAINT `fk_game_play_game` FOREIGN KEY (`game_oid`) REFERENCES
`game` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `game_play_action`
--

DROP TABLE IF EXISTS `game_play_action`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `game_play_action` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `score` decimal(19,2) DEFAULT NULL,
  `player_oid` int(11) DEFAULT NULL,
  `role_oid` int(11) DEFAULT NULL,
  `game_play_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_game_play_action_player` (`player_oid`),
```

```
  KEY `fk_game_play_action_role` (`role_oid`),
  KEY `fk_game_play_action_game_play` (`game_play_oid`),
  CONSTRAINT `fk_game_play_action_game_play` FOREIGN KEY
(`game_play_oid`) REFERENCES `game_play` (`oid`),
  CONSTRAINT `fk_game_play_action_player` FOREIGN KEY (`player_oid`)
REFERENCES `player` (`user_oid`),
  CONSTRAINT `fk_game_play_action_role` FOREIGN KEY (`role_oid`)
REFERENCES `role` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gamestats`
--

DROP TABLE IF EXISTS `gamestats`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gamestats` (
  `oid` int(11) NOT NULL,
  `hoursplayed` decimal(19,2) DEFAULT NULL,
  `score` decimal(19,2) DEFAULT NULL,
  `game_oid` int(11) DEFAULT NULL,
  `player_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_gamestats_game` (`game_oid`),
  KEY `fk_gamestats_player` (`player_oid`),
  CONSTRAINT `fk_gamestats_game` FOREIGN KEY (`game_oid`) REFERENCES
`game` (`oid`),
  CONSTRAINT `fk_gamestats_player` FOREIGN KEY (`player_oid`)
REFERENCES `player` (`user_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `player`
--

DROP TABLE IF EXISTS `player`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `player` (
  `user_oid` int(11) NOT NULL,
  `oid` int(11) NOT NULL,
  `nickname` varchar(255) DEFAULT NULL,
  `avatar` varchar(255) DEFAULT NULL,
```

```
  `reputation` int(11) DEFAULT NULL,
  `player_type` varchar(255) DEFAULT NULL,
  `player_level` int(11) DEFAULT NULL,
  `experience_points` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`user_oid`),
  KEY `fk_player_user` (`user_oid`),
  CONSTRAINT `fk_player_user` FOREIGN KEY (`user_oid`) REFERENCES
`user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `question`
--

DROP TABLE IF EXISTS `question`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `question` (
  `oid` int(11) NOT NULL,
  `topic` varchar(255) DEFAULT NULL,
  `level` int(11) DEFAULT NULL,
  `text` varchar(255) DEFAULT NULL,
  `closed_ended` bit(1) DEFAULT NULL,
  `utility_question` bit(1) DEFAULT NULL,
  `utility_name` varchar(255) DEFAULT NULL,
  `quiz_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_question_quiz` (`quiz_oid`),
  CONSTRAINT `fk_question_quiz` FOREIGN KEY (`quiz_oid`) REFERENCES
`quiz` (`game_oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `question_instance`
--

DROP TABLE IF EXISTS `question_instance`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `question_instance` (
  `oid` int(11) NOT NULL,
  `date` date DEFAULT NULL,
  `score` decimal(19,2) DEFAULT NULL,
```

```
  `guessed` bit(1) DEFAULT NULL,
  `question_oid` int(11) DEFAULT NULL,
  `player_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_question_instance_question` (`question_oid`),
  KEY `fk_question_instance_player` (`player_oid`),
  CONSTRAINT `fk_question_instance_player` FOREIGN KEY
(`player_oid`) REFERENCES `player` (`user_oid`),
  CONSTRAINT `fk_question_instance_question` FOREIGN KEY
(`question_oid`) REFERENCES `question` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `quiz`
--

DROP TABLE IF EXISTS `quiz`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `quiz` (
  `game_oid` int(11) NOT NULL,
  `oid` int(11) NOT NULL,
  PRIMARY KEY (`game_oid`),
  KEY `fk_quiz_game` (`game_oid`),
  CONSTRAINT `fk_quiz_game` FOREIGN KEY (`game_oid`) REFERENCES
`game` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `role`
--

DROP TABLE IF EXISTS `role`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `role` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `game_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_role_game` (`game_oid`),
  CONSTRAINT `fk_role_game` FOREIGN KEY (`game_oid`) REFERENCES
`game` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `role_ability`
--

DROP TABLE IF EXISTS `role_ability`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `role_ability` (
  `role_oid` int(11) NOT NULL,
  `ability_oid` int(11) NOT NULL,
  PRIMARY KEY (`role_oid`,`ability_oid`),
  KEY `fk_role_ability_role` (`role_oid`),
  KEY `fk_role_ability_ability` (`ability_oid`),
  CONSTRAINT `fk_role_ability_ability` FOREIGN KEY (`ability_oid`)
REFERENCES `ability` (`oid`),
  CONSTRAINT `fk_role_ability_role` FOREIGN KEY (`role_oid`)
REFERENCES `role` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `user`
--

DROP TABLE IF EXISTS `user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user` (
  `oid` int(11) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `birth_date` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
```

```
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2015-04-30 17:45:13
```

## 8.3 Gamification engine subschema

```
-- MySQL dump 10.13  Distrib 5.6.17, for Win32 (x86)
--
-- Host: localhost    Database: community_new_newdata
-- ------------------------------------------------------
-- Server version 5.6.23-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014      SET      @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101                SET                @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `action_instance`
--

DROP TABLE IF EXISTS `action_instance`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `action_instance` (
  `oid` int(11) NOT NULL,
  `executor` varchar(255) DEFAULT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `score` decimal(19,2) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `tag` varchar(255) DEFAULT NULL,
  `link` varchar(255) DEFAULT NULL,
  `rank_oid` int(11) DEFAULT NULL,
  `action_type_oid` int(11) DEFAULT NULL,
```

```
  `object_key` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_action_instance_rank` (`rank_oid`),
  KEY `idx_action_instance_action_typ` (`action_type_oid`),
  CONSTRAINT     `fk_action_instance_action_type`     FOREIGN     KEY
(`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT   `fk_action_instance_rank`   FOREIGN   KEY   (`rank_oid`)
REFERENCES `community_user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
--       Temporary       table       structure       for       view
`action_instance_action_area_vi`
--

DROP TABLE IF EXISTS `action_instance_action_area_vi`;
/*!50001 DROP VIEW IF EXISTS `action_instance_action_area_vi`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `action_instance_action_area_vi` (
  `oid` tinyint NOT NULL,
  `der_attr` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Temporary table structure for view `action_instance_daily_vi`
--

DROP TABLE IF EXISTS `action_instance_daily_vi`;
/*!50001 DROP VIEW IF EXISTS `action_instance_daily_vi`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `action_instance_daily_vi` (
  `action_type_oid` tinyint NOT NULL,
  `date` tinyint NOT NULL,
  `daily_occurrence` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Temporary table structure for view `action_instance_name_view`
--
```

```
DROP TABLE IF EXISTS `action_instance_name_view`;
/*!50001 DROP VIEW IF EXISTS `action_instance_name_view`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `action_instance_name_view` (
  `oid` tinyint NOT NULL,
  `der_attr` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `action_type`
--

DROP TABLE IF EXISTS `action_type`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `action_type` (
  `oid` int(11) NOT NULL,
  `check_time_elapsed` tinyint(1) DEFAULT NULL,
  `milliseconds_time_elapsed` int(11) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `repeatable` tinyint(1) DEFAULT NULL,
  `score` decimal(19,2) DEFAULT NULL,
  `reputation` tinyint(1) DEFAULT NULL,
  `participation` tinyint(1) DEFAULT NULL,
  `area` varchar(255) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `gamified_application_oid` int(11) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  `thematic_area_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_action_type_gamified_appli` (`gamified_application_oid`),
  KEY `fk_action_type_thematic_area` (`thematic_area_oid`),
  CONSTRAINT     `fk_action_type_gamified_applic`     FOREIGN     KEY
(`gamified_application_oid`)    REFERENCES    `gamified_application`
(`oid`),
  CONSTRAINT     `fk_action_type_thematic_area`     FOREIGN     KEY
(`thematic_area_oid`) REFERENCES `thematic_area` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `badge_action`
--
```

```
DROP TABLE IF EXISTS `badge_action`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `badge_action` (
  `badge_type_oid` int(11) NOT NULL,
  `action_type_oid` int(11) NOT NULL,
  PRIMARY KEY (`badge_type_oid`,`action_type_oid`),
  KEY `idx_badge_action_badge_type` (`badge_type_oid`),
  KEY `idx_badge_action_action_type` (`action_type_oid`),
  CONSTRAINT `fk_badge_action_action_type` FOREIGN KEY
(`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT `fk_badge_action_badge_type` FOREIGN KEY
(`badge_type_oid`) REFERENCES `badge_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `badge_instance`
--

DROP TABLE IF EXISTS `badge_instance`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `badge_instance` (
  `oid` int(11) NOT NULL,
  `score` decimal(19,2) DEFAULT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `rank_oid` int(11) DEFAULT NULL,
  `badge_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_badge_instance_rank` (`rank_oid`),
  KEY `idx_badge_instance_badge_type` (`badge_type_oid`),
  CONSTRAINT `fk_badge_instance_badge_type` FOREIGN KEY
(`badge_type_oid`) REFERENCES `badge_type` (`oid`),
  CONSTRAINT `fk_badge_instance_rank` FOREIGN KEY (`rank_oid`)
REFERENCES `community_user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `badge_type`
--

DROP TABLE IF EXISTS `badge_type`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `badge_type` (
  `oid` int(11) NOT NULL,
  `area` varchar(255) DEFAULT NULL,
  `needed_score` decimal(19,2) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `hd_image` varchar(255) DEFAULT NULL,
  `key` varchar(255) DEFAULT NULL,
  `importance` int(11) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `checked_image` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `hd_checked_image` varchar(255) DEFAULT NULL,
  `sort_number` int(11) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  `image_2` varchar(255) DEFAULT NULL,
  `imageblob` longblob,
  `hd_image_2` varchar(255) DEFAULT NULL,
  `hd_imageblob` longblob,
  `checked_image_2` varchar(255) DEFAULT NULL,
  `checked_imageblob` longblob,
  `hd_checked_image_2` varchar(255) DEFAULT NULL,
  `hd_checked_imageblob` longblob,
  `thematic_area_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_badge_type_thematic_area` (`thematic_area_oid`),
  CONSTRAINT      `fk_badge_type_thematic_area`      FOREIGN      KEY
(`thematic_area_oid`) REFERENCES `thematic_area` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Temporary table structure for view `badgeimportancebyuser`
--

DROP TABLE IF EXISTS `badgeimportancebyuser`;
/*!50001 DROP VIEW IF EXISTS `badgeimportancebyuser`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `badgeimportancebyuser` (
  `badge_instance` tinyint NOT NULL,
  `user` tinyint NOT NULL,
  `nickname_area` tinyint NOT NULL,
  `importance` tinyint NOT NULL
```

```
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Temporary table structure for view `badgetype_sortco`
--

DROP TABLE IF EXISTS `badgetype_sortco`;
/*!50001 DROP VIEW IF EXISTS `badgetype_sortco`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `badgetype_sortco` (
  `oid` tinyint NOT NULL,
  `der_attr` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `bundle_data`
--

DROP TABLE IF EXISTS `bundle_data`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `bundle_data` (
  `oid` int(11) NOT NULL,
  `key` varchar(255) DEFAULT NULL,
  `locale` varchar(255) DEFAULT NULL,
  `message` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `common_data`
--

DROP TABLE IF EXISTS `common_data`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `common_data` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `area` varchar(255) DEFAULT NULL,
```

```
  `hd_image` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `community_user`
--

DROP TABLE IF EXISTS `community_user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `community_user` (
  `oid` int(11) NOT NULL,
  `company_name` varchar(255) DEFAULT NULL,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  `city` varchar(255) DEFAULT NULL,
  `forum_level` int(11) DEFAULT NULL,
  `small_photo` varchar(255) DEFAULT NULL,
  `twitter` varchar(255) DEFAULT NULL,
  `country` varchar(255) DEFAULT NULL,
  `public_profile` tinyint(1) DEFAULT NULL,
  `geographical_area` varchar(255) DEFAULT NULL,
  `website` varchar(255) DEFAULT NULL,
  `big_photo` varchar(255) DEFAULT NULL,
  `bio` text,
  `linkedin` varchar(255) DEFAULT NULL,
  `certification_level` int(11) DEFAULT NULL,
  `kb_level` int(11) DEFAULT NULL,
  `store_level` int(11) DEFAULT NULL,
  `participation_monthly` decimal(19,2) DEFAULT NULL,
  `forum_badge` varchar(255) DEFAULT NULL,
  `certification_badge` varchar(255) DEFAULT NULL,
  `kb_badge` varchar(255) DEFAULT NULL,
  `store_badge` varchar(255) DEFAULT NULL,
  `kb_badge_title` varchar(255) DEFAULT NULL,
  `store_badge_title` varchar(255) DEFAULT NULL,
  `forum_badge_title` varchar(255) DEFAULT NULL,
  `certification_badge_title` varchar(255) DEFAULT NULL,
  `birthdate` date DEFAULT NULL,
  `participation` decimal(19,2) DEFAULT NULL,
  `credit` decimal(19,2) DEFAULT NULL,
  `facebook` varchar(255) DEFAULT NULL,
  `google` varchar(255) DEFAULT NULL,
```

```
  `iso_code` varchar(255) DEFAULT NULL,
  `small_photo_2` varchar(255) DEFAULT NULL,
  `small_photoblob` longblob,
  `big_photo_2` varchar(255) DEFAULT NULL,
  `big_photoblob` longblob,
  `registration_date` datetime DEFAULT NULL,
  `latitude` decimal(19,6) DEFAULT NULL,
  `longitude` decimal(19,6) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  CONSTRAINT `fk_rank_usertable` FOREIGN KEY (`oid`) REFERENCES
`user` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
--       Temporary       table       structure       for       view
`community_user_credits_availab`
--

DROP TABLE IF EXISTS `community_user_credits_availab`;
/*!50001 DROP VIEW IF EXISTS `community_user_credits_availab`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `community_user_credits_availab` (
  `oid` tinyint NOT NULL,
  `der_attr` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
--       Temporary       table       structure       for       view
`community_user_credits_spent_v`
--

DROP TABLE IF EXISTS `community_user_credits_spent_v`;
/*!50001 DROP VIEW IF EXISTS `community_user_credits_spent_v`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `community_user_credits_spent_v` (
  `oid` tinyint NOT NULL,
  `der_attr` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
```

```
-- Table structure for table `containers_mail`
--

DROP TABLE IF EXISTS `containers_mail`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `containers_mail` (
  `oid` int(11) NOT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `text` text,
  `alias` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gamified_application`
--

DROP TABLE IF EXISTS `gamified_application`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gamified_application` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `api_key` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `gamifiedapplication_thematic_a`
--

DROP TABLE IF EXISTS `gamifiedapplication_thematic_a`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `gamifiedapplication_thematic_a` (
  `gamified_application_oid` int(11) NOT NULL,
  `thematic_area_oid` int(11) NOT NULL,
  PRIMARY KEY (`gamified_application_oid`,`thematic_area_oid`),
  KEY `fk_gamifiedapplication_themati` (`gamified_application_oid`),
  KEY `fk_gamifiedapplication_thema_2` (`thematic_area_oid`),
  CONSTRAINT     `fk_gamifiedapplication_thema_2`     FOREIGN     KEY
(`thematic_area_oid`) REFERENCES `thematic_area` (`oid`),
```

```
  CONSTRAINT     `fk_gamifiedapplication_themati`     FOREIGN     KEY
(`gamified_application_oid`)     REFERENCES     `gamified_application`
(`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `geographical_area`
--

DROP TABLE IF EXISTS `geographical_area`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `geographical_area` (
  `oid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `goal`
--

DROP TABLE IF EXISTS `goal`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `goal` (
  `oid` int(11) NOT NULL,
  `title` varchar(255) DEFAULT NULL,
  `completion_date` date DEFAULT NULL,
  `community_user_user_id` int(11) DEFAULT NULL,
  `badge_type_oid` int(11) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  `consumption` decimal(19,2) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `fk_goal_community_user` (`community_user_user_id`),
  KEY `fk_goal_badge_type` (`badge_type_oid`),
  CONSTRAINT  `fk_goal_badge_type`  FOREIGN  KEY  (`badge_type_oid`)
REFERENCES `badge_type` (`oid`),
  CONSTRAINT       `fk_goal_community_user`       FOREIGN       KEY
(`community_user_user_id`) REFERENCES `community_user` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```
-- Table structure for table `goal_action_type`
--

DROP TABLE IF EXISTS `goal_action_type`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `goal_action_type` (
  `goal_oid` int(11) NOT NULL,
  `action_type_oid` int(11) NOT NULL,
  PRIMARY KEY (`goal_oid`,`action_type_oid`),
  KEY `fk_goal_action_type_goal` (`goal_oid`),
  KEY `fk_goal_action_type_action_typ` (`action_type_oid`),
  CONSTRAINT     `fk_goal_action_type_action_typ`     FOREIGN     KEY
(`action_type_oid`) REFERENCES `action_type` (`oid`),
  CONSTRAINT  `fk_goal_action_type_goal`  FOREIGN  KEY  (`goal_oid`)
REFERENCES `goal` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `group_moduletable`
--

DROP TABLE IF EXISTS `group_moduletable`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `group_moduletable` (
  `groupoid` int(11) NOT NULL,
  `moduleoid` int(11) NOT NULL,
  PRIMARY KEY (`groupoid`,`moduleoid`),
  KEY `idx_group_moduletable_grouptab` (`groupoid`),
  KEY `idx_group_moduletable_siteview` (`moduleoid`),
  CONSTRAINT     `fk_group_moduletable_grouptabl`     FOREIGN     KEY
(`groupoid`) REFERENCES `grouptable` (`oid_2`),
  CONSTRAINT     `fk_group_moduletable_siteviewt`     FOREIGN     KEY
(`moduleoid`) REFERENCES `siteviewtable` (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `grouptable`
--

DROP TABLE IF EXISTS `grouptable`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `grouptable` (
  `oid_2` int(11) NOT NULL,
  `groupname` varchar(255) DEFAULT NULL,
  `siteviewoid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid_2`),
  KEY `idx_grouptable_siteviewtable` (`siteviewoid`),
  CONSTRAINT `fk_grouptable_siteviewtable` FOREIGN KEY
(`siteviewoid`) REFERENCES `siteviewtable` (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Temporary table structure for view
`headquarter_user_partecipation`
--

DROP TABLE IF EXISTS `headquarter_user_partecipation`;
/*!50001 DROP VIEW IF EXISTS `headquarter_user_partecipation`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `headquarter_user_partecipation` (
  `oid` tinyint NOT NULL,
  `partecipation` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Temporary table structure for view
`headquarter_user_participation_monthly`
--

DROP TABLE IF EXISTS `headquarter_user_participation_monthly`;
/*!50001 DROP VIEW IF EXISTS
`headquarter_user_participation_monthly`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `headquarter_user_participation_monthly` (
  `oid` tinyint NOT NULL,
  `participation_monthly` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Temporary table structure for view
`headquarter_user_participation_seven_days`
--
```

```
DROP TABLE IF EXISTS `headquarter_user_participation_seven_days`;
/*!50001 DROP VIEW IF EXISTS
`headquarter_user_participation_seven_days`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `headquarter_user_participation_seven_days` (
  `oid` tinyint NOT NULL,
  `participation_seven_days` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `job_blob_triggers`
--

DROP TABLE IF EXISTS `job_blob_triggers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_blob_triggers` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `BLOB_DATA` blob,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT `JOB_BLOB_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`,
`TRIGGER_NAME`, `TRIGGER_GROUP`) REFERENCES `job_triggers`
(`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_calendars`
--

DROP TABLE IF EXISTS `job_calendars`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_calendars` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `CALENDAR_NAME` varchar(200) NOT NULL,
  `CALENDAR` blob NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`CALENDAR_NAME`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_cron_triggers`
--

DROP TABLE IF EXISTS `job_cron_triggers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_cron_triggers` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `CRON_EXPRESSION` varchar(120) NOT NULL,
  `TIME_ZONE_ID` varchar(80) DEFAULT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT `JOB_CRON_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`,
`TRIGGER_NAME`,     `TRIGGER_GROUP`)     REFERENCES     `job_triggers`
(`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_fired_triggers`
--

DROP TABLE IF EXISTS `job_fired_triggers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_fired_triggers` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `ENTRY_ID` varchar(95) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `INSTANCE_NAME` varchar(200) NOT NULL,
  `FIRED_TIME` bigint(13) NOT NULL,
  `PRIORITY` int(11) NOT NULL,
  `STATE` varchar(16) NOT NULL,
  `JOB_NAME` varchar(200) DEFAULT NULL,
  `JOB_GROUP` varchar(200) DEFAULT NULL,
  `IS_NONCONCURRENT` varchar(1) DEFAULT NULL,
  `REQUESTS_RECOVERY` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`SCHED_NAME`,`ENTRY_ID`),
  KEY `IDX_JOB_FT_TRIG_INST_NAME` (`SCHED_NAME`,`INSTANCE_NAME`),
```

```
  KEY                                   `IDX_JOB_FT_INST_JOB_REQ_RCVRY`
(`SCHED_NAME`,`INSTANCE_NAME`,`REQUESTS_RECOVERY`),
  KEY `IDX_JOB_FT_J_G` (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_FT_JG` (`SCHED_NAME`,`JOB_GROUP`),
  KEY                                                   `IDX_JOB_FT_T_G`
(`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `IDX_JOB_FT_TG` (`SCHED_NAME`,`TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_job_details`
--

DROP TABLE IF EXISTS `job_job_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_job_details` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `JOB_NAME` varchar(200) NOT NULL,
  `JOB_GROUP` varchar(200) NOT NULL,
  `DESCRIPTION` varchar(250) DEFAULT NULL,
  `JOB_CLASS_NAME` varchar(250) NOT NULL,
  `IS_DURABLE` varchar(1) NOT NULL,
  `IS_NONCONCURRENT` varchar(1) NOT NULL,
  `IS_UPDATE_DATA` varchar(1) NOT NULL,
  `REQUESTS_RECOVERY` varchar(1) NOT NULL,
  `JOB_DATA` blob,
  PRIMARY KEY (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_J_REQ_RECOVERY` (`SCHED_NAME`,`REQUESTS_RECOVERY`),
  KEY `IDX_JOB_J_GRP` (`SCHED_NAME`,`JOB_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_locks`
--

DROP TABLE IF EXISTS `job_locks`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_locks` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `LOCK_NAME` varchar(40) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`LOCK_NAME`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_paused_trigger_grps`
--

DROP TABLE IF EXISTS `job_paused_trigger_grps`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_paused_trigger_grps` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_scheduler_state`
--

DROP TABLE IF EXISTS `job_scheduler_state`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_scheduler_state` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `INSTANCE_NAME` varchar(200) NOT NULL,
  `LAST_CHECKIN_TIME` bigint(13) NOT NULL,
  `CHECKIN_INTERVAL` bigint(13) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`INSTANCE_NAME`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_simple_triggers`
--

DROP TABLE IF EXISTS `job_simple_triggers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_simple_triggers` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `REPEAT_COUNT` bigint(7) NOT NULL,
```

```
  `REPEAT_INTERVAL` bigint(12) NOT NULL,
  `TIMES_TRIGGERED` bigint(10) NOT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT `JOB_SIMPLE_TRIGGERS_ibfk_1` FOREIGN KEY (`SCHED_NAME`,
`TRIGGER_NAME`,     `TRIGGER_GROUP`)     REFERENCES     `job_triggers`
(`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_simprop_triggers`
--

DROP TABLE IF EXISTS `job_simprop_triggers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `job_simprop_triggers` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `STR_PROP_1` varchar(512) DEFAULT NULL,
  `STR_PROP_2` varchar(512) DEFAULT NULL,
  `STR_PROP_3` varchar(512) DEFAULT NULL,
  `INT_PROP_1` int(11) DEFAULT NULL,
  `INT_PROP_2` int(11) DEFAULT NULL,
  `LONG_PROP_1` bigint(20) DEFAULT NULL,
  `LONG_PROP_2` bigint(20) DEFAULT NULL,
  `DEC_PROP_1` decimal(13,4) DEFAULT NULL,
  `DEC_PROP_2` decimal(13,4) DEFAULT NULL,
  `BOOL_PROP_1` varchar(1) DEFAULT NULL,
  `BOOL_PROP_2` varchar(1) DEFAULT NULL,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  CONSTRAINT      `JOB_SIMPROP_TRIGGERS_ibfk_1`      FOREIGN      KEY
(`SCHED_NAME`,     `TRIGGER_NAME`,     `TRIGGER_GROUP`)     REFERENCES
`job_triggers` (`SCHED_NAME`, `TRIGGER_NAME`, `TRIGGER_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `job_triggers`
--

DROP TABLE IF EXISTS `job_triggers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `job_triggers` (
  `SCHED_NAME` varchar(120) NOT NULL,
  `TRIGGER_NAME` varchar(200) NOT NULL,
  `TRIGGER_GROUP` varchar(200) NOT NULL,
  `JOB_NAME` varchar(200) NOT NULL,
  `JOB_GROUP` varchar(200) NOT NULL,
  `DESCRIPTION` varchar(250) DEFAULT NULL,
  `NEXT_FIRE_TIME` bigint(13) DEFAULT NULL,
  `PREV_FIRE_TIME` bigint(13) DEFAULT NULL,
  `PRIORITY` int(11) DEFAULT NULL,
  `TRIGGER_STATE` varchar(16) NOT NULL,
  `TRIGGER_TYPE` varchar(8) NOT NULL,
  `START_TIME` bigint(13) NOT NULL,
  `END_TIME` bigint(13) DEFAULT NULL,
  `CALENDAR_NAME` varchar(200) DEFAULT NULL,
  `MISFIRE_INSTR` smallint(2) DEFAULT NULL,
  `JOB_DATA` blob,
  PRIMARY KEY (`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`),
  KEY `SCHED_NAME` (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_T_J` (`SCHED_NAME`,`JOB_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_T_JG` (`SCHED_NAME`,`JOB_GROUP`),
  KEY `IDX_JOB_T_C` (`SCHED_NAME`,`CALENDAR_NAME`),
  KEY `IDX_JOB_T_G` (`SCHED_NAME`,`TRIGGER_GROUP`),
  KEY `IDX_JOB_T_STATE` (`SCHED_NAME`,`TRIGGER_STATE`),
  KEY                                          `IDX_JOB_T_N_STATE`
(`SCHED_NAME`,`TRIGGER_NAME`,`TRIGGER_GROUP`,`TRIGGER_STATE`),
  KEY                                        `IDX_JOB_T_N_G_STATE`
(`SCHED_NAME`,`TRIGGER_GROUP`,`TRIGGER_STATE`),
  KEY `IDX_JOB_T_NEXT_FIRE_TIME` (`SCHED_NAME`,`NEXT_FIRE_TIME`),
  KEY                                            `IDX_JOB_T_NFT_ST`
(`SCHED_NAME`,`TRIGGER_STATE`,`NEXT_FIRE_TIME`),
  KEY                                       `IDX_JOB_T_NFT_MISFIRE`
(`SCHED_NAME`,`MISFIRE_INSTR`,`NEXT_FIRE_TIME`),
  KEY                                    `IDX_JOB_T_NFT_ST_MISFIRE`
(`SCHED_NAME`,`MISFIRE_INSTR`,`NEXT_FIRE_TIME`,`TRIGGER_STATE`),
  KEY                                 `IDX_JOB_T_NFT_ST_MISFIRE_GRP`
(`SCHED_NAME`,`MISFIRE_INSTR`,`NEXT_FIRE_TIME`,`TRIGGER_GROUP`,`TRIG
GER_STATE`),
  CONSTRAINT   `JOB_TRIGGERS_ibfk_1`   FOREIGN   KEY   (`SCHED_NAME`,
`JOB_NAME`, `JOB_GROUP`) REFERENCES `job_job_details` (`SCHED_NAME`,
`JOB_NAME`, `JOB_GROUP`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Temporary table structure for view `max_date_action_instance`
--
```

```
DROP TABLE IF EXISTS `max_date_action_instance`;
/*!50001 DROP VIEW IF EXISTS `max_date_action_instance`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `max_date_action_instance` (
  `action_type_oid` tinyint NOT NULL,
  `rank_oid` tinyint NOT NULL,
  `maxDate` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Temporary table structure for view `mostimportant_badge`
--

DROP TABLE IF EXISTS `mostimportant_badge`;
/*!50001 DROP VIEW IF EXISTS `mostimportant_badge`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `mostimportant_badge` (
  `oid` tinyint NOT NULL,
  `rankoid` tinyint NOT NULL,
  `area` tinyint NOT NULL,
  `title` tinyint NOT NULL,
  `importance` tinyint NOT NULL,
  `checked_image_2` tinyint NOT NULL,
  `checked_imageblob` tinyint NOT NULL,
  `hd_checked_image_2` tinyint NOT NULL,
  `hd_checked_imageblob` tinyint NOT NULL,
  `sort_number` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `notification`
--

DROP TABLE IF EXISTS `notification`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `notification` (
  `oid` int(11) NOT NULL,
  `creation_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
```

```
  `code` varchar(255) DEFAULT NULL,
  `status` varchar(255) DEFAULT NULL,
  `delivery_date` timestamp NULL DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `rank_oid` int(11) DEFAULT NULL,
  `reward_type_oid` int(11) DEFAULT NULL,
  `text_mail_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_notification_rank` (`rank_oid`),
  KEY `idx_notification_reward_type` (`reward_type_oid`),
  KEY `idx_notification_text_mail` (`text_mail_oid`),
  CONSTRAINT `fk_notification_rank` FOREIGN KEY (`rank_oid`)
REFERENCES `community_user` (`oid`),
  CONSTRAINT `fk_notification_reward_type` FOREIGN KEY
(`reward_type_oid`) REFERENCES `reward_type` (`oid`),
  CONSTRAINT `fk_notification_text_mail` FOREIGN KEY
(`text_mail_oid`) REFERENCES `text_mail` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `reward_instance`
--

DROP TABLE IF EXISTS `reward_instance`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `reward_instance` (
  `oid` int(11) NOT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `score` decimal(19,2) DEFAULT NULL,
  `rank_oid` int(11) DEFAULT NULL,
  `reward_type_oid` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_reward_instance_rank` (`rank_oid`),
  KEY `idx_reward_instance_reward_typ` (`reward_type_oid`),
  CONSTRAINT `fk_reward_instance_rank` FOREIGN KEY (`rank_oid`)
REFERENCES `community_user` (`oid`),
  CONSTRAINT `fk_reward_instance_reward_type` FOREIGN KEY
(`reward_type_oid`) REFERENCES `reward_type` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `reward_type`
```

```
--

DROP TABLE IF EXISTS `reward_type`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `reward_type` (
  `oid` int(11) NOT NULL,
  `needed_points` decimal(19,2) DEFAULT NULL,
  `available` tinyint(1) DEFAULT NULL,
  `image` varchar(255) DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `description` text,
  `image_2` varchar(255) DEFAULT NULL,
  `imageblob` longblob,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `siteviewtable`
--

DROP TABLE IF EXISTS `siteviewtable`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `siteviewtable` (
  `oid_2` int(11) NOT NULL,
  `moduledomainname` varchar(255) DEFAULT NULL,
  `siteviewid` varchar(255) DEFAULT NULL,
  `modulename` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `text_chunk`
--

DROP TABLE IF EXISTS `text_chunk`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `text_chunk` (
  `oid` int(11) NOT NULL,
  `languagecode` varchar(255) DEFAULT NULL,
```

```
  `key` varchar(255) DEFAULT NULL,
  `message` text,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `text_mail`
--

DROP TABLE IF EXISTS `text_mail`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `text_mail` (
  `oid` int(11) NOT NULL,
  `code` varchar(255) DEFAULT NULL,
  `language_code` varchar(255) DEFAULT NULL,
  `body` text,
  `description` varchar(255) DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `containers_oid_header` int(11) DEFAULT NULL,
  `containers_oid_footer` int(11) DEFAULT NULL,
  PRIMARY KEY (`oid`),
  KEY `idx_text_mail_containers_mail` (`containers_oid_header`),
  KEY `idx_text_mail_containers_mai_2` (`containers_oid_footer`),
  CONSTRAINT      `fk_text_mail_containers_mail`      FOREIGN      KEY
(`containers_oid_header`) REFERENCES `containers_mail` (`oid`),
  CONSTRAINT     `fk_text_mail_containers_mail_2`     FOREIGN     KEY
(`containers_oid_footer`) REFERENCES `containers_mail` (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `thematic_area`
--

DROP TABLE IF EXISTS `thematic_area`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `thematic_area` (
  `oid` int(11) NOT NULL,
  `area_name` varchar(255) DEFAULT NULL,
  `checked_image` varchar(255) DEFAULT NULL,
  `hd_image` varchar(255) DEFAULT NULL,
  `hd_checked_image` varchar(255) DEFAULT NULL,
```

```
  `checked_imageblob` blob,
  `hd_checked_imageblob` blob,
  `hd_imageblob` blob,
  PRIMARY KEY (`oid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `user`
--

DROP TABLE IF EXISTS `user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user` (
  `user_id` int(11) NOT NULL,
  `email` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `internal` tinyint(1) DEFAULT NULL,
  `username` varchar(255) DEFAULT NULL,
  `groupoid` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  KEY `idx_usertable_grouptable` (`groupoid`),
  CONSTRAINT  `fk_usertable_grouptable`  FOREIGN  KEY  (`groupoid`)
REFERENCES `grouptable` (`oid_2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `user_grouptable`
--

DROP TABLE IF EXISTS `user_grouptable`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_grouptable` (
  `useroid` int(11) NOT NULL,
  `groupoid` int(11) NOT NULL,
  PRIMARY KEY (`useroid`,`groupoid`),
  KEY `idx_user_grouptable_usertable` (`useroid`),
  KEY `idx_user_grouptable_grouptable` (`groupoid`),
  CONSTRAINT     `fk_user_grouptable_grouptable`     FOREIGN     KEY
(`groupoid`) REFERENCES `grouptable` (`oid_2`),
  CONSTRAINT  `fk_user_grouptable_usertable`  FOREIGN  KEY  (`useroid`)
REFERENCES `user` (`user_id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Temporary table structure for view `user_information`
--

DROP TABLE IF EXISTS `user_information`;
/*!50001 DROP VIEW IF EXISTS `user_information`*/;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
/*!50001 CREATE TABLE `user_information` (
  `oid` tinyint NOT NULL,
  `country` tinyint NOT NULL,
  `area_geografica` tinyint NOT NULL,
  `small_photo` tinyint NOT NULL,
  `big_photo` tinyint NOT NULL,
  `first_name` tinyint NOT NULL,
  `last_name` tinyint NOT NULL,
  `twitter` tinyint NOT NULL,
  `linkedin` tinyint NOT NULL,
  `website` tinyint NOT NULL,
  `bio` tinyint NOT NULL,
  `city` tinyint NOT NULL,
  `company_name` tinyint NOT NULL,
  `email` tinyint NOT NULL,
  `internal` tinyint NOT NULL
) ENGINE=MyISAM */;
SET character_set_client = @saved_cs_client;

--
-- Final view structure for view `action_instance_action_area_vi`
--

/*!50001 DROP TABLE IF EXISTS `action_instance_action_area_vi`*/;
/*!50001 DROP VIEW IF EXISTS `action_instance_action_area_vi`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `action_instance_action_area_vi` AS select `al1`.`oid`
```

```
AS `oid`,`al2`.`area` AS `der_attr` from (`action_instance` `al1`
left join `action_type` `al2` on((`al1`.`action_type_oid` =
`al2`.`oid`))) */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `action_instance_daily_vi`
--

/*!50001 DROP TABLE IF EXISTS `action_instance_daily_vi`*/;
/*!50001 DROP VIEW IF EXISTS `action_instance_daily_vi`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `action_instance_daily_vi` AS select
`action_instance`.`action_type_oid` AS
`action_type_oid`,cast(`action_instance`.`date` as date) AS
`date`,count(0) AS `daily_occurrence` from `action_instance` group
by `action_instance`.`action_type_oid`,cast(`action_instance`.`date`
as date) order by
`action_instance`.`action_type_oid`,cast(`action_instance`.`date` as
date) */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `action_instance_name_view`
--

/*!50001 DROP TABLE IF EXISTS `action_instance_name_view`*/;
/*!50001 DROP VIEW IF EXISTS `action_instance_name_view`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
```

```
/*!50001 VIEW `action_instance_name_view` AS select `al1`.`oid` AS
`oid`,`al2`.`name` AS `der_attr` from (`action_instance` `al1` left
join     `action_type`     `al2`     on((`al1`.`action_type_oid`     =
`al2`.`oid`))) */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `badgeimportancebyuser`
--

/*!50001 DROP TABLE IF EXISTS `badgeimportancebyuser`*/;
/*!50001 DROP VIEW IF EXISTS `badgeimportancebyuser`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001  VIEW  `badgeimportancebyuser`  AS  select  `m`.`oid`  AS
`badge_instance`,`q`.`oid`       AS       `user`,`c`.`area`       AS
`nickname_area`,max(`c`.`importance`)     AS     `importance`     from
((`badge_type` `c` join `badge_instance` `m`) join `community_user`
`q`)  where  ((`m`.`badge_type_oid`  =  `c`.`oid`)  and  (`q`.`oid`  =
`m`.`rank_oid`)) group by `q`.`oid`,`c`.`area` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `badgetype_sortco`
--

/*!50001 DROP TABLE IF EXISTS `badgetype_sortco`*/;
/*!50001 DROP VIEW IF EXISTS `badgetype_sortco`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001   VIEW   `badgetype_sortco`   AS   select   `al1`.`oid`   AS
```

```
`oid`,(`al2`.`sort_number` or `al3`.`sort_number`) AS `der_attr`
from ((`badge_type` `al1` join `badge_type` `al2`) join `badge_type`
`al3`) where ((`al2`.`key` = 'area') and (`al1`.`area` =
`al2`.`area`) and (`al3`.`key` = 'level') and (`al1`.`importance` =
`al3`.`sort_number`)) */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `community_user_credits_availab`
--

/*!50001 DROP TABLE IF EXISTS `community_user_credits_availab`*/;
/*!50001 DROP VIEW IF EXISTS `community_user_credits_availab`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `community_user_credits_availab` AS select `al1`.`oid`
AS `oid`,(case when isnull((`al1`.`credit` - `al2`.`der_attr`)) then
0 else (`al1`.`credit` - `al2`.`der_attr`) end) AS `der_attr` from
(`community_user` `al1` left join `community_user_credits_spent_v`
`al2` on((`al1`.`oid` = `al2`.`oid`))) */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `community_user_credits_spent_v`
--

/*!50001 DROP TABLE IF EXISTS `community_user_credits_spent_v`*/;
/*!50001 DROP VIEW IF EXISTS `community_user_credits_spent_v`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `community_user_credits_spent_v` AS select `al1`.`oid`
```

```
AS `oid`,(case when isnull(sum(`al2`.`score`)) then 0 else
sum(`al2`.`score`) end) AS `der_attr` from (`community_user` `al1`
left join `reward_instance` `al2` on((`al1`.`oid` =
`al2`.`rank_oid`))) group by `al1`.`oid` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `headquarter_user_partecipation`
--

/*!50001 DROP TABLE IF EXISTS `headquarter_user_partecipation`*/;
/*!50001 DROP VIEW IF EXISTS `headquarter_user_partecipation`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `headquarter_user_partecipation` AS select `al1`.`oid`
AS `oid`,sum(`al2`.`score`) AS `partecipation` from
((`community_user` `al1` join `action_instance` `al2`) join
`action_type` `al3`) where ((`al3`.`participation` = 1) and
(`al1`.`oid` = `al2`.`rank_oid`) and (`al2`.`action_type_oid` =
`al3`.`oid`)) group by `al1`.`oid` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view
`headquarter_user_participation_monthly`
--

/*!50001 DROP TABLE IF EXISTS
`headquarter_user_participation_monthly`*/;
/*!50001 DROP VIEW IF EXISTS
`headquarter_user_participation_monthly`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
```

```
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `headquarter_user_participation_monthly` AS select `r`.`oid` AS `oid`,sum(`al2`.`score`) AS `participation_monthly` from (`action_instance` `al2` join `community_user` `r` on((`r`.`oid` = `al2`.`rank_oid`))) where ((month(`al2`.`date`) = month(now())) and (year(`al2`.`date`) = year(now()))) group by `r`.`oid` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
--        Final        view        structure        for        view `headquarter_user_participation_seven_days`
--

/*!50001          DROP          TABLE          IF          EXISTS `headquarter_user_participation_seven_days`*/;
/*!50001          DROP          VIEW          IF          EXISTS `headquarter_user_participation_seven_days`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `headquarter_user_participation_seven_days` AS select `r`.`oid` AS `oid`,sum(`al3`.`score`) AS `participation_seven_days` from ((`community_user` `r` left join `action_instance` `al3` on((`r`.`oid` = `al3`.`rank_oid`))) left join `action_type` `al4` on((`al3`.`action_type_oid` = `al4`.`oid`))) where ((`al3`.`date` <= now()) and (`al3`.`date` >= (now() - interval 7 day)) and (`al4`.`participation` = 1)) group by `r`.`oid` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `max_date_action_instance`
--

/*!50001 DROP TABLE IF EXISTS `max_date_action_instance`*/;
/*!50001 DROP VIEW IF EXISTS `max_date_action_instance`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
```

```
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001      VIEW      `max_date_action_instance`      AS      select
`action_instance`.`action_type_oid`                               AS
`action_type_oid`,`action_instance`.`rank_oid`                    AS
`rank_oid`,max(`action_instance`.`date`)     AS     `maxDate`     from
`action_instance`                        group                      by
`action_instance`.`action_type_oid`,`action_instance`.`rank_oid` */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `mostimportant_badge`
--

/*!50001 DROP TABLE IF EXISTS `mostimportant_badge`*/;
/*!50001 DROP VIEW IF EXISTS `mostimportant_badge`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001       VIEW       `mostimportant_badge`       AS       select
`badge`.`badge_instance`        AS        `oid`,`rr`.`oid`        AS
`rankoid`,`dict`.`area`       AS       `area`,`dict`.`title`       AS
`title`,`badge`.`importance`                                      AS
`importance`,`dict`.`checked_image_2`                             AS
`checked_image_2`,`dict`.`checked_imageblob`                      AS
`checked_imageblob`,`dict`.`hd_checked_image_2`                   AS
`hd_checked_image_2`,`dict`.`hd_checked_imageblob`                AS
`hd_checked_imageblob`,`dict`.`sort_number`  AS  `sort_number`  from
((`badgeimportancebyuser`  `badge`  join  `community_user`  `rr`)  join
`badge_type`       `dict`)       where       ((`dict`.`area`       =
`badge`.`nickname_area`)       and       (`dict`.`importance`       =
`badge`.`importance`) and (`rr`.`oid` = `badge`.`user`)) */;
/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
```

```
-- Final view structure for view `user_information`
--

/*!50001 DROP TABLE IF EXISTS `user_information`*/;
/*!50001 DROP VIEW IF EXISTS `user_information`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8 */;
/*!50001 SET character_set_results     = utf8 */;
/*!50001 SET collation_connection      = utf8_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `user_information` AS select `r1`.`oid` AS
`oid`,`r1`.`country` AS `country`,`r1`.`geographical_area` AS
`area_geografica`,`r1`.`small_photo` AS
`small_photo`,`r1`.`big_photo` AS `big_photo`,`r1`.`first_name` AS
`first_name`,`r1`.`last_name` AS `last_name`,`r1`.`twitter` AS
`twitter`,`r1`.`linkedin` AS `linkedin`,`r1`.`website` AS
`website`,`r1`.`bio` AS `bio`,`r1`.`city` AS
`city`,`r1`.`company_name` AS `company_name`,`c1`.`email` AS
`email`,`c1`.`internal` AS `internal` from (`community_user` `r1`
join `user` `c1` on((`c1`.`user_id` = `r1`.`oid`))) where
(`r1`.`public_profile` = 1) */;

/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2015-04-30 17:52:09
```