# sa.engine Whitepaper

Stream Analyze Sweden AB
Version 2.1
2020-07-02
sa_whitepaper_2.1.pdf

**sa.engine** is a platform and extensible toolbox by Stream Analyze Sweden for easy and flexible development and deployment of applications that process and analyze real-time streams of data in distributed and mobile environments. Streams that are produced by sensors on mobile or edge devices can be analyzed on-line interactively. An edge device can be, e.g., an Android unit, a desktop computer, a Raspberry Pi, or MCUs like MangOH Red or ARM M4. sa.engine has a very small footprint (from 100kB to 5MB depending on configuration) and is hardware and OS agnostic and fully independent of any third party software.

The combination in sa.engine of a main-memory database, a computational engine, a data stream processor, and an inference engine implemented on edge devices allows **edge analytics** directly on the edge devices rather than the contemporary approach of first uploading all data from devices to a server and then do all processing centrally on a server or in the cloud. The approach allows for drastic data reduction by filtering the data streams already in the devices and only population analyses over collections of devices need to be made centrally.

On server computers, in order to collect data from edges and combine and forward the data to other server-based systems and users, sa.engine systems can also be configured as **stream servers**. For example, whenever the analysis model in some edge device detects strong vibrations by doing a computation over the latest readings from its accelerometer, a data stream containing the frequency spectrum of the vibrations along with the geographical position of the device is transmitted to a stream server. If the stream server receives many such streams at the same time from devices in geographical positions close to each other, it may indicate an earthquake. The stream server may furthermore forward the received data to other systems, e.g. for permanent central storage or batch analysis.

To analyze streaming data interactively on a very high and user-oriented level, sa.engine allows analysts and engineers to develop computations, filters and transformations over data using a **query anguage** called **OSQL** (Object Stream Query Language). With OSQL computations and filters over real-time streaming data are defined as mathematical formulas and expressions, called **stream models**. A stream model is a set of definitions of mathematical functions, filters, and other expressions over a stream of measurements. With OSQL the models can be specified on a very high level without deep programming knowledge; you specify **what** to do rather than writing detailed programs expressing **how** to execute the models. The user needs not worry about details on how to efficiently implement programs that integrate large numbers of edge devices and servers.

An OSQL query that continuously makes computations or filtering over measurements in a data stream is called a **continuous query, CQ**. sa.engine allows analysts to interactively specify CQs for continuously analyzing measurements flowing through edge devices and stream servers in real-time. The result of a CQ is a real-time **object stream** of processed and filtered measurements, for example a CQ returning a stream of the position vectors of a device measured every second when it is close to a given geo-position.

Both stream models and user data are stored in very fast object-oriented **in-memory databases** residing on the edge devices and in the stream servers. These databases are important for data stream processing, which usually involves matching in real-time fast flowing stream objects against data in a database. For example, to determine that the frequency spectrum of a measured vibration may later destroy an edge device, the frequencies measured by vibration sensors on the device is matched against a local database of resonance frequencies in the device.

The system includes a library of over 1000 predefined OSQL-functions for math/stat computations, object stream filtering and transformation, signal processing, model and data management, and much more. The function library is continuously extended for new customer needs and it is easy to define and deploy new user functions on-the-fly.

In order to combine object streams from several edges, sa.engine supports **fusion queries** that combine object streams. An example of a fusion query is a CQ observing when several edge devices in an area detect strong vibrations at the same time. The user is alerted when the fusion query produces results, perhaps together with a visualization in real-time of the magnitude of the observed vibrations. A user can then interactively send new queries on-the-fly to affected devices to find out details of their causes.

Fusion queries require the integrated data streams to be comparable even though the involved object streams may represent the same or similar data in different ways. For example, one device may represent temperature in Fahrenheit while another one use Celsius. To be able to combine such **heterogeneous** data streams from different devices, sa.engine allows **mediator** models to be defined as queries and functions that harmonize arriving heterogeneous object streams by transforming them to a universal model, called an **ontology**, in stream servers that integrate data streams from different edges. Mediation can be, e.g., mapping local names of sensors to a universally known nomenclature, measurement unit conversions, and calibrations of local measurements.

Existing algorithms and libraries implemented in a regular programming language, such as C or Java, can be plugged into the system as *foreign OSQL functions* using programming language specific APIs. The foreign functions can then be transparently used in OSQL queries and expressions. For example, a large library of basic mathematical, statistical, and machine learning algorithms are implemented as foreign OSQL functions in C.

In order to access external data streams, sa.engine provides **wrapper** functionality, which is APIs that enable mapping over incoming data stream objects as they arrive in order to inject them into the sa.engine kernel so that the accessed stream can be used in CQs expressed in OSQL. Different streams often represent their elements using different physical data structures, so the wrapper functionality includes the ability to extend sa.engine with new physical data representations and to convert the external data representations to a format already supported by the system. The wrappers are defined as OSQL functions that return object streams from wrapped data sources. There is a library of predefined wrappers to interoperate with common data infrastructures such as relational databases through JDBC and data processing systems through Kafka, Azure IoT Hub, or MQTT. New wrappers can easily be developed.

Common machine learning algorithms such as DBSCAN, DenStream, k-NN, and k-means are available as predefined OSQL models and the user can easily extend this with other algorithms defined in OSQL or as foreign functions. Learning and inference are supported on both edges and servers.

Machine learning requires pre-processing of sensor data before applying the learned inference algorithm, followed by post-processing of the inferred knowledge. With sa.engine both pre and post-processing are easily expressed using the powerful object stream filtering, transformation and math/stat analytics capabilities of OSQL.

The system includes a deep learning subsystem, SANN, where neural network models defined by Tensorflow/Tensorboard can be automatically translated into an internal binary SANN representation and pushed out to the edge databases. Once stored in an edge database the SANN inference engine can analyze local sensor readings to detect anomalies.

SANN furthermore allows to continue training and even building neural networks on edge devices. Thus, centrally trained models can be further retrained and modified on edge devices to adapt their behavior to their environment.