Key-Value Pairs and NoSQL Databases: A Novel Concept to Manage Biologging Data in Data Repositories

Holger Dettki[‡], Debora Arlt[‡], Johan Bäckman[§], Mathieu Blanchet[§]

‡ Swedish University of Agricultural Sciences, Uppsala, Sweden
§ Lund University, Lund, Sweden

Corresponding author: Holger Dettki (holger.dettki@slu.se)

Abstract

Traditional data scheme concepts for biologging data previously relied on traditional relational databases and fixed normalized tables. In practice, this means that a repository contains separate, fixed table structures for each type of sensor data. Prominent examples are the current Wireless Remote Animal Monitoring (WRAM) data schema or the now discontinued ZoaTrack approach. While the traditional approach worked fine as long as few sensors with fixed data types were used, rapid technological development continuously introduces new sensor types and more advanced sensor platforms. This means more data providers, new data types, and rapidly increasing amounts of data. Storage solutions using relational data models generate constant requirements for additional tables, changes to existing table structures, and as a consequence, changes to the overall data scheme in the repository. Further, it becomes very difficult to adapt to emerging international standards, as any change in a particular data field in a single table may have wide ranging consequences to the overall database structure.

A concept better suited to deal with the growing amount of sensors and sensor types is the Key-Value Pair (KVP) concept: A KVP is a data type that includes two pieces of data that have a group of key identifiers and a set of associated values. The KVP concept has been used for a long time in data exchange/transport (e.g., JavaScript Object Notation (JSON), XML). Today, very good database solutions exist (e.g., MongoDB, Apache Cassandra DB, Apache HBase) that use KVP directly as the data store. Within a KVP, there are two related data elements. The first element, the key, is a constant used to identify the data type. The other element is a value, which is a variable representing the actual measurement of the data type. In other words, instead of using two separate tables with different table structures to store data e.g., from an acceleration sensor and a GPS-sensor (Global Positioning System), we simply define key-IDs representing the different data types of a GPS-sensor and store its associated measurement values, for example: longitude, latitude, date, and time. We can then store the key-IDs for 'Acceleration' in the same table

with it's associated unique values without requiring any change to the overall data scheme. (Fig. 1).

The data is stored in a key-value store: a non-relational or NoSQL database specifically designed to handle key-value pairs. The obvious advantage is flexibility: A key-value store allows any new sensor type easily to be added to the repository without requiring any structural change. Furthermore, this concept allows for scalability, speed, and optimization of storage space. While the traditional concept required the input of ‘null’ for optional values, key-value stores just skip this particular optional value, resulting in smaller storage requirements. Biologging datasets also differ from more classical 'biodiversity' datasets in size: a single standard 3-axis-acceleration sensor measuring at 30 Hz (30 measurements per second) produces ca. one billion measurements per axis and year for a single individual. Thus, high scalability is a necessity when serving modern sensor systems that accumulate these vast amounts of data. Databases like MongoDB are easy to design as distributed systems. High performance comes from the flexible data structures, e.g., the possibility of storing large structures of data in a single document, which allows performance-critical queries to be made in a single request, but also from the horizontal scalability, which allows for load distribution across multiple hardware systems.

In 2021 the former CAnMove (Center for Animal Movement) initiative at Lund University, Sweden, which previously adapted the ZoaTrack application for Swedish needs, and the WRAM biotelemetry e-infrastructure at the Swedish University of Agricultural Sciences (SLU) joined forced within the Swedish Biodiversity Data Infrastructure (SBDI) to develop a new data model based on the KVP-concept.

We started analyzing the data and sensor types used in the WRAM and CAnMove repositories and constructed KVPs that can cover all current data. We also added metadata descriptions for projects, datasets and sensors used (Fig. 2). The concept is currently being tested with an implementation into MongoDB.

While different tables are used to identify the project, dataset or sampling event in a one-to-many relationship (Fig. 2), the KVP table ‘Record’ contains the actual measurements. In order to identify which sensor can take which measurements, the KVP table ‘Sensor’ serves as a ‘look-up’ table. Hence, to add new sensors types to the repository, only records in the KPV table ‘Sensor’ have to be added to update the repository to handle and store these data. Data in a KVP model are easy to parse and since we strictly use open standards when available, such as Darwin Core in our data, it is relatively easy to publish and exchange data in other formats.

As NoSQL databases are now mature products with many proven use cases, there is no reason to hesitate building production systems for biologging repositories based on these. Further work will be done to ensure coherence with the emerging standards for biologging data to enable seamless data sharing across other biologging repositories, such as Moveb ank, and data aggregation into the Global Biodiversity Information Facility (GBIF).

Keywords

data schema, data storage, data modelling, performance, scalability

Presenting author

Holger Dettki

Presented at

TDWG 2023

Conflicts of interest

The authors have declared that no competing interests exist.

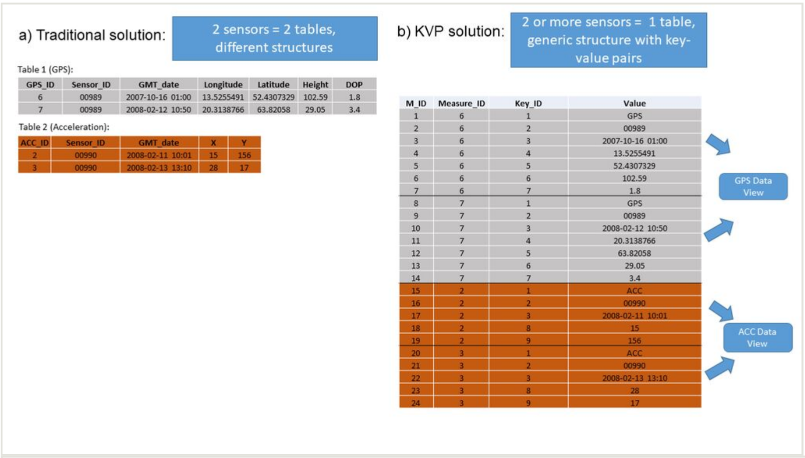
a) Traditional solution:
2 sensors = 2 tables, different structures
b) KVP solution:
2 or more sensors = 1 table, generic structure with key-value pairs
GPS Data View
ACC Data View

Table 1 (GPS):

GPS_ID	Sensor_ID	GMT_date	Longitude	Latitude	Height	DOP
6	00989	2007-10-16 01:00	13.5255491	52.4307329	102.59	1.8
7	00989	2008-02-12 10:50	20.3138766	63.82058	29.05	3.4

Table 2 (Acceleration):

ACC_ID	Sensor_ID	GMT_date	X	Y
2	00990	2008-02-11 10:01	15	156
3	00990	2008-02-13 13:10	28	17

M_ID	Measure_ID	Key_ID	Value
1	6	1	GPS
2	6	2	00989
3	6	3	2007-10-16 01:00
4	6	4	13.5255491
5	6	5	52.4307329
6	6	6	102.59
7	6	7	1.8
8	7	1	GPS
9	7	2	00989
10	7	3	2008-02-12 10:50
11	7	4	20.3138766
12	7	5	63.82058
13	7	6	29.05
14	7	7	3.4
15	2	1	ACC
16	2	2	00990
17	2	3	2008-02-11 10:01
18	2	8	15
19	2	9	156
20	3	1	ACC
21	3	2	00990
22	3	3	2008-02-13 13:10
23	3	8	28
24	3	9	17

Figure 1.

Two tables from a relational database (a) are represented in a non-relational key-value store as Key-Value Pairs (KVP:s) (b)

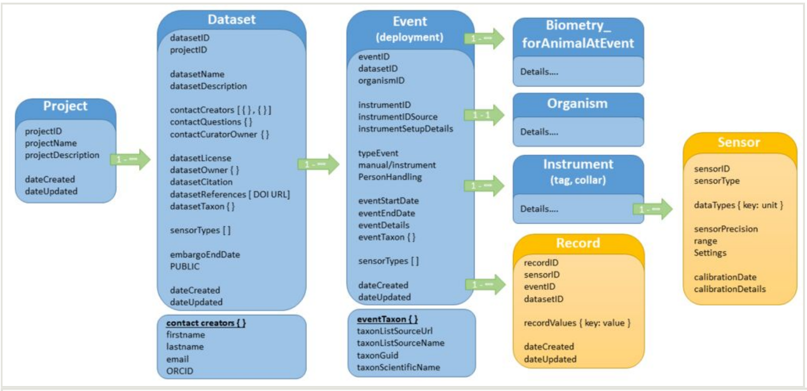

Figure 2.

Concept of the key-value store with the different KVP tables