# Research Statement and Agenda

Samuel Madden
Computer Science Division
University of California, Berkeley
madden@cs.berkeley.edu

In my research, I have proposed, built, and evaluated a declarative query processor for wireless, battery-powered sensor networks, and explored the research challenges, such as designing appropriate language abstractions and devising query optimization and execution techniques, that arise in building such a system. Each node in a sensor network is a small, radio-equipped, battery powered computer that carries some set of sensors which provide information about the physical environment – for example, light levels, acceleration, or ambient gas concentration. These networks are increasingly being deployed in remote locations to monitor and respond to events; for instance, scientists have placed a sensor network on Great Duck Island (off the coast of Maine) to monitor the occupancy of nests of Storm Petrels, a type of endangered sea-bird traditionally monitored by physical observation.

My work on sensor networks fits within a general theme for my research on building adaptive systems for extracting and processing data from distributed, continuous, time-varying data sources, or *streams*. Sensor data is one example of a data stream, although streams arise in many contexts: stock quotes and news feeds from the web, logs from routers and servers, and phone call records from telecommunications companies. As a member of the Telegraph project at UC Berkeley, I have explored general issues related to query processing over streaming data sources, especially focusing on adaptivity to changing rates and data distributions within those streams over time.

## Research Approach

My research is strongly systems-oriented, with an emphasis on building real systems and using them to evaluate software architectures and research concepts. I believe it is crucial to evaluate software systems through real implementations, since the complexity of modern distributed systems can be extremely hard to capture through simulation alone. Furthermore, the software artifacts which result from such implementations are an essential contribution of systems-oriented research that drive both product development and future research.

My research reflects this philosophy: the TinyDB query processor, which I built while working at Intel-Research, Berkeley, is a real software system for query processing in sensor networks. The resulting system is being incorporated into several research projects within Intel Corporation's research division, and I provide support to a growing community of TinyDB users. I have also been actively involved in the development of the TinyOS operating system for sensor networks and in the Telegraph project, where I worked as part of a team to build an several adaptive query processing systems that have been deployed and are publicly distributed.

## A Declarative Query System for Sensor Networks

Developing the software for sensor network deployments has traditionally been an extremely time consuming task: for example, it takes a skilled graduate student several weeks to write the low-level embedded code which runs on sensors in a deployment like Great Duck Island. TinyDB dramatically reduces the time to develop such deployments: a typical monitoring application consists of a few statements in a simple query language. Such statements specify a set of sensor attributes of interest, a data collection rate, and a set of filters and aggregates to apply to data as it flows out of the network. Users input queries on a PC, where the TinyDB front end converts them into an execution plan and distributes that plan throughout the sensor network. Each device collects the appropriate data, processes and combines it with data from neighboring nodes, and returns the results back to user's PC. TinyDB manages many of the subtle difficulties that arise in sensor network programming: for example, it provides a robust routing layer that can adapt to network failures, the ability to dynamically add nodes to a running network, a scheduler

that puts sensors into a low-power sleep mode when there is no data to collect or packets to route, and a metadata management system that allows users to introspect on the capabilities of devices and easily install software handlers for new sensing hardware. In addition to these substantial benefits that TinyDB offers users of sensor networks, my work has addressed several fundamental research challenges that arose while building the system:

**In-Network Aggregation for Sensor Networks**: Aggregation operators in database systems combine a set of values together to produce a new aggregate value representing some summary of that set; typical database systems include aggregates like minimum, average, and sum. In the TAG (Tiny AGgregation) system, I proposed and studied a distributed framework for computing aggregates within a sensor network. Pushing aggregates into a network of devices is desirable because such operations can significantly reduce the amount of data that must be transmitted; this is particularly important in sensor networks as communication tends to dominate power consumption. The major innovation of TAG is a set of communication-reducing optimizations whose applicability depends on the semantics of the aggregation operators in a particular query. To allow the system to determine when these techniques can be applied, TAG is based on a simple taxonomy that maps from functional properties of aggregates to specific optimizations. This provides a basis for extensibility: new, user-defined aggregation functions can be installed, and, as long as they are classified in this taxonomy, can be automatically optimized by the system.

**Acquisitional Query Processing**: In traditional database systems, data is provided *a priori*, and the goal of the database system is to process that data once it is inside of the boundaries of the query engine. In contrast, in acquisitional query processing (ACQP), I proposed that the query processor take an active role in the physical acquisition of the (sensor) data. That is, it manages which devices sample data for a particular query, the order in which samples are acquired, and how the system combines the sampling of sensors with the application of traditional query processing operators. By controlling the locations and costs of acquiring data in this way, ACQP systems significantly reduce power consumption compared to traditional passive systems.

TinyDB incorporates a number of acquisitional features and techniques designed to minimize power consumption and communication. As an example of a simple acquisitional technique, consider the physical sampling of sensor devices, which can consume significant amounts of power. In a traditional database system, those samples would be captured outside of the DBMS and query processing operators would be applied to a complete tuple representing the state of all the sensors at some time. In contrast, TinyDB interleaves sampling of sensors and application of operators, applying selection and aggregation operators to partially-complete tuples as soon as possible. If these operators filter out a particular tuple, the cost of sampling the remaining fields in that tuple can be saved; properly ordering sampling of sensors can save as much power as idling the processor when there is no work to be done.

## Continuous and Adaptive Query Processing

Users of sensor networks typically wish to stream the data into a PC-based database. Unfortunately, the interfaces of traditional databases are not entirely appropriate for querying such data streams, since they provide one-shot queries that give query answers at a particular point in time rather than a continuous stream of answers updated as new readings enter the database. Thus, the other major focus of my work has been on building systems that facilitate the processing of data streams and the application of *continuous queries* to such streams. Answering these queries is fundamentally different from answering conventional queries because answers must be produced online (since queries potentially never terminate) rather than in a batch at the end of query execution. My work in this area has also included research on *adaptive query processing*, where the order of operators applied to data may change from one data tuple to the next rather than being fixed at the time of query execution (as in a traditional DBMS). Adaptivity is important in query processing environments where statistics used to choose the order of operators may be inaccurate or change over the duration of a query.

The Telegraph and TelegraphCQ projects, which I was involved in as a part of the UC Berkeley Database Group, explored continuous and adaptive query processing over a variety of data sources. The design of the more recent TelegraphCQ project is based largely upon my research on Continuously Adaptive Continuous Queries (CACQ).

**Continuously Adaptive Continuous Queries:** CACQ identified the significant opportunities for adaptive query processing in continuous queries. Because continuous queries run for long periods of time, they are especially amenable to adaptive techniques, since optimizer estimates made at the start of queries are likely to become invalid over the lifetime of the query. CACQ also presented a novel technique, called *tuple lineage*, for sharing work between similar queries over one or more common data streams. Such sharing is particularly important in sensor applications where there may be large numbers of users observing and querying the same sensor streams. For example, freeways in California have been instrumented with inductive-loop sensors that register vehicles which pass overhead; during peak commuter times, one could imagine tens of thousands of outstanding and similar queries over that data as travelers monitor delays along their planned routes. To support this number of simultaneous queries, it is essential for query processing systems over large, real-time data sets such as these to exploit possibilities for sharing between related queries; CACQ was a substantial step in this direction.

## Future Research

My thesis research has raised a number of potential areas for future research in the area of query processing in sensor networks. I also believe there is great promise in the application of similar data processing techniques to other large distributed environments beyond sensors.

In the near term, there are significant areas for work related to my thesis. One of the simplifying assumptions in my work to date is that the sensors in a sensor network are homogeneous; as more and more real deployments of such networks occur, it is clear that this will not be the case. Some sensors will be relatively resource impoverished, with small batteries and limited storage, while others will be much more powerful, with connections to the power grid, large external memories, or access to fast networks. Understanding how to take advantage of such asymmetries by, for example, routing data through nodes with more power or faster networks, is an important area for future research that I have only begun to explore. A related area for future work has to do with choosing which parts of queries to push down into sensor networks and which to execute on powered basestations where users receive results. As deployments of TinyDB that are currently underway come online, I expect that a number of other issues of this sort will arise; an important part of my future research plan is to follow up on these deployments and extend and maintain TinyDB to address the new requirements that they present.

Looking further out, there are opportunities to apply some of the lessons from my research on sensor networks to other domains. For example, one of the most exciting features of TinyDB is its ability to exploit query and operator semantics to significantly reduce communication and alter the routes which data takes as it flows through an ad-hoc network. For instance, when a sensor network is answering a query which applies only to some subset of the individual devices (e.g. nodes in a particular geographic region), a significant reduction in the number of active nodes can be achieved by having sensors route data through neighbors that are also providing data relevant to the query. This savings is possible because those neighbors would be active even if they were not forwarding data for other nodes. Minimizing the number of active nodes is essential for power conservation purposes. Simple, declarative queries make this sort of routing optimization possible, since queries do not specify paths data must take from one part of the network to the next. Such semantically-driven optimizations are certainly possible in other highly distributed environments like the Internet and peer-to-peer systems; identifying specific instances will require an understanding of the unusual hardware characteristics and query workloads that arise in these environments.

More generally, the intersection between databases, distributed systems, and networks is a rich area for future research. The simplicity and semantically well-defined nature of declarative database languages provides a powerful way to specify the data of interest in complex systems that are large, lossy, or subject to high variability in rates or availability of resources, just as TinyDB makes interactions with sensor networks vastly simpler than conventional approaches. On the other hand, just as the query language and types of operators used in TinyDB are shaped by the specific needs and applications of users of sensor networks, query processors will have to be broadened and enriched to handle the unusual challenges of these new environments.