

590BDL Water Quality Database project  
Dan Wright  
12 Dec 2006

## **Background**

The goal of this database project is to produce a shared water quality resource database for the state of Illinois. There are 30+ data collection projects and outstanding small databases currently scattered around the state of Illinois (unpublished data); examples include Great Rivers, RiverWatch, and Friends of the Fox. Most of these databases are electronic, but in formats that are not web-accessible; some of them are still maintained solely on paper. This document is intended to report current status of efforts thus far, and to propose database schema and standards for future development. In particular, I hope that having comprehensive documentation available will make it easier to acquire appropriate funding and talent for the implementation phase of the project.

## **Current Status**

There is a database front-end and table structure available from a previous effort at developing a water quality database. The front-end has not been evaluated in detail yet, and that will probably be beyond the ultimate scope of this particular effort. The motivation for reworking the existing database schema is the acknowledged need for extensibility and flexibility as primary features of biodiversity data resources, especially when primary data may be re-used for other purposes [1]. A secondary motivation is that the current database is not running anywhere at the moment, and it was felt that a redesign would ultimately be a

better use of project time than re-installing the previous software.

The current design is tightly focused and structured around two particular data collection efforts – RiverWatch and Great Rivers – and is therefore not robust with regards to extensibility (to more surveys) and flexibility (with regards to data types and data acquisition protocols). In particular, accommodating an arbitrary number of surveys and data collection protocols is not possible under the existing system without adding tables to the database for each survey/protocol combination. This would be unwieldy and hard to manage after only a few such table additions. An ER diagram of the existing design is in Appendix B.

The front end for the database system currently consists of a PHP, CSS, and JavaScript interface to the MySQL backend. A number of important features, like automatic data validation via AJAX-style form controls, have been written and it appears that the system as it stands could be gotten running with a minimum of effort. Doing so is beyond the scope of this project, however, and will require a non-trivial investment of time that would be better put into updating the system for future extensibility.

For reference, the project report for this first version is included as Appendix C.

## **Proposed design**

Appendix A contains an updated database design that should be more suitable for the long-term needs of the project. The most significant changes are:

- A survey is defined in a table that links it to protocols; each survey no longer has its own table;
- The protocol is tied to a survey, not to a collector;
- Each survey can have more than one protocol; each data collection event specifies which

protocol was used.

The design rationale has been to create a system that intended to support long-term usage of a single database to store water quality survey results for all of Illinois. In principle, there is no reason that this database could not manage an unlimited number of water quality surveys nation-wide, though in practice this would probably not be feasible due to administrative overhead and resource constraints. However, producing more databases of this design in other states and federating data across them would be reasonable and facilitate data sharing and aggregation to support large-scale water quality projects.

### *Self-describing interface*

One goal of the new database design is to facilitate the generalization of survey interface code across *all* of the surveys managed by the database. This will provide a significant benefit to maintainability of the interface and funding of the database as a whole, because much less programmer time will be needed to keep up the front end.

Each survey is associated with a particular data collection protocol. These protocols consist of an elaboration of the types of measurements to be collected under the protocol, along with flags for presentation order (in a data collection interface) and a flag telling the interface if a particular measurement is mandatory under the protocol being elaborated. Under the protocol, every measurement type contains a complete description of the parameters needed to present the data in a user interface: min/max allowed values, details of the method used to collect it, and so on. A complete data input interface can be built on-the-fly from these descriptions, which provides a number of benefits:

- The same interface building code can be used for all surveys managed with this

database;

- Changing collection protocols can be done solely by updating the database, without needed to change user-interface code at all;
- Adding new data types and collection methods requires no changes to the user-interface code;
- Managing collection protocols and measurements can be done by scientific personnel with no knowledge of the underlying data structure, if an appropriate interface is provided for adding protocols and their associated data types;
- Adding new surveys can be done with only minimal knowledge of web programming skills – copying the files containing the user-interface code from another survey and changing branding information (like the logo) should be all that is required.

This kind of management simplicity should serve the kind of survey projects that will use this database very well, since they are usually minimally funded and need to spend what funding they have on data collection instrumentation and a few professional staff to manage volunteers.

### Data quality

A vital component of any field data collection exercise is close attention to the quality of the collected data. This is of particular concern in the case of the surveys being used as models for this database project, because the majority of the data is collected by Citizen Scientist volunteers rather than comprehensively-trained professionals [2]. The use of volunteers in data collection is not a problem *per se*, as long as proper steps are taken to record the level of expertise of the data collector and the precision of the measurement, and to ensure that this

information is preserved and can be presented along with the data in any reporting interface. In this way, users can always make their own evaluation of the fitness of the data for their particular use. Indeed, fitness of use is the ultimate determinant of data quality – data may be suitable for some purposes but not others depending on a range of factors, and it is vital to preserve sufficient information for making these fitness judgments [3].

According to Chapman [4], the keys to the improvement of data quality are the prevention and correction of errors. This database project is focused on the prevention of errors for two reasons: first, Chapman also states that “error prevention is considered to be far superior to error detection, since detection is often costly and can never guarantee to be 100% successful” (p. 8). Second is a matter of practicality: the database and data entry interface can be structured to automatically enforce certain constraints on data as it is input, but the kind of systematic manual intervention needed to correct error after-the-fact is beyond the scope of this project to specify. However, after a working system is built some attention should be given to this issue, paying particular attention to the issues of data cleaning mentioned by Chapman.

Error prevention is clearly the focus of the surveys that are currently collecting data, as articulated by the RiverWatch manual [2]. The focus on providing training and documentation to volunteers to insure maximum accuracy in the field is reflected in the database design, where the training level of a collector is documented and tracked over time. In this way, a user of the reporting functionality of the database can be sure of the relative expertise of the data collectors and make decisions about re-use of the data. This kind of understanding of data confidence and accuracy becomes particularly important when combining data sets from multiple sites, as would be the case in the nationwide data federation scenario mentioned above.

To facilitate the goal of error prevention, the database supports the storage of allowed and expected ranges for numerical data so that bounds checking and verification can be done on-the-fly, either in the user interface or in the database itself. These kind of checks were implemented in the previous version of the database, and should be carried over to any future efforts.

## **Data Sharing and Federation**

Since data sharing is a vital part of any ecological/biodiversity informatics project, some thought has been given to standards-based approaches to this problem. The most relevant standard is the Ecological Markup Language (EML; <http://knb.ecoinformatics.org/software/eml/>), which defines a set of XML schema and metadata for the storage and transport of ecological data. EML supports the storage of data collection protocols and other meta-information, in addition to the data itself; thus, it is a natural fit for representing this content of the water quality database project.

The most sensible approach would be to write export filters that would make the database content available in EML for any outside entities. The capability to export any report from the database in EML should be implemented, along with the capability to export the entire database wholesale into a set of EML files. These EML exports could be used for data backup and archiving, in addition to facilitating interoperability. Due to the technical complexity of such format translation programs, however, it is recommended to approach these as a secondary feature to be implemented after the core functionality of the database – data input, storage, and reporting – is finished.

## Usage Scenarios

This section will walk through the usage scenarios for three basic user types: volunteer citizen-scientist, survey manager, and data consumer.

### Citizen-scientist

First, we will follow Dick, an environmentally concerned resident of Champaign, during a data collection exercise for RiverWatch at the Vermilion river. He needs to collect three data points: nitrogen level, temperature, and pH. He has a tablet PC equipped with a cellular GPRS modem, and will be doing live data input to the database system.

When he arrives at the site, he gets out all the equipment he will need to take his measurements: pH paper, thermometer, and some sort of thing that determines the nitrogen level. He then boots up the tablet PC and opens the data input web page for RiverWatch. After logging in with his email address and password, the system generates a data input form for Dick to use, based on the “RiverWatch volunteer Vermilion River” data collection protocol. He first selects his location from a drop-down list of locations he has collected data at previously, and chooses “Monthly data collection site for Dick”.

The data collection protocol specifies that pH, temperature, and nitrogen levels must all be measured, along with a visual evaluation of river cloudiness. In the database, there are several methods for measuring pH: electronic meter, pH paper, and titration. However, Dick has only been trained on the use of pH paper, so the web interface only presents him with this option for collection method. If he had additional training, he would be able to select which methodology he used at this particular data collection event. The data input form he sees

contains three numerical input fields for pH, temperature, and nitrogen level; the temperature and nitrogen level blanks also specify their units, “degrees C” and “PPM” respectively. In addition, there is a set of radio buttons for stream clarity; the options available in this part of the form are “Clear”, “Cloudy”, and “Other”. The “Other” option has an associated text box for Dick to fill in a description of stream condition if it does not match one of the other available options.

Before he begins collecting data, Dick realizes that he has forgotten where exactly he should take the measurements. He clicks on a link to the data collection protocol that is listed on his data input form; this link opens a new web browser window that brings up the protocol definition, including the detailed collection methodology for each data point. After reviewing this information, Dick feels confident about his collection exercise and proceeds to the riverbank to take his readings. He first takes the pH reading, and enters “7” into the form. Proceeding to temperature, he enters “70”, having mistakenly read the Fahrenheit side of the thermometer. The input form pops up a dialog box asking, “This value is outside the expected range! Please double check your reading!”. Dick takes another look and realizes his error, and changes the reading to “21”. For the nitrogen level reading, he does [something] and gets a reading of [something]. When inputting the data, he accidentally puts a “-” in front of his reading and is prompted that this is not an allowed value; a PPM reading  $< 0$  is not possible. He quickly fixes the typo and moves on, selecting “Cloudy” for the stream clarity. Once he finishes filling out the form, he clicks “Save” and is taken to another screen that displays the data he just input and asks him to confirm that it is correct. He quickly reviews the data, and clicks the “Confirm” button to store the data in the database. Finished, he packs up his tablet PC and measurement tools and heads home.

## Survey Manager

Jane is a manager at the RiverWatch survey, and they have recently changed to a new data collection protocol. She must go into the RiverWatch water quality database and add the new collection protocol. Jane logs into the RiverWatch survey site, she is presented with more options than Dick was above; she is designated in the system as a “survey\_admin”, and so she is able to do things like edit data after it has been entered, and create new protocols and measurement types.

Today Jane is interested in creating a new protocol for RiverWatch, so she follows the link that takes her to a protocol setup form. She is presented with a form asking for a name and a description for this protocol, so she fills these values in with “New RiverWatch” and a long description of the protocol. She takes time to document this carefully, as this – along with the collection method data associated with each measurement type – is what her field volunteers (like Dick) will see when they follow the “Help” link on their data input forms. She then clicks on a button labeled “Add measures to this protocol” which takes her to a screen with a drop-down list of available measures, and buttons labeled “Add this measure”, “Add new measure”, and “Done adding measures”. She first selects “Temperature” from the drop down box and clicks “Add this measure”. Since there are several measurement types named “Temperature” in the database, she is presented with a pop-up window with the text “Please select which 'Temperature' measures you need” and a list with check-boxes next to each kind of “Temperature”. A kind of temperature is defined by listing the “Units” and “MeasurementMethod” for each kind. She checks the boxes next to the kind using “degrees C” and “mercury thermometer”, and the kind using “degrees C” and “digital thermometer”, then clicks a button labeled “Continue”. The pop-up then closes, and the main window loads a

screen very similar to the first one, but with the measures already added listed at the top of the screen. She follows the same procedure to add “pH” (collected via pH meter or pH paper) and “nitrogen level”. Jane needs to add a new measure to the database: “oxygen concentration”, measured with a new electronic meter RiverWatch has just acquired. She clicks the button “Add new measure”, and is taken to a screen where she fills in the name (“oxygen concentration”; this is the second measure of this name in the database), measurement method, etc. After adding the measure, it is added to the growing protocol. She then adds a measure for “water clarity” (a multi-select data type already present in the database), and then clicks “Done adding measures”.

The system then presents a summary screen displaying the entire protocol, and gives her the option to edit what has been input, or save the whole thing. Jane reviews her work and clicks “Save”. She is then asked, “Should this protocol be made the primary protocol for RiverWatch?”, with two buttons labeled “yes” and “no”. Since Jane wants all of her field volunteers to begin using the new protocol immediately, she clicks “yes”. The next time Dick is in the field, he automatically presented with a data input form for this new protocol.

### Data Consumer

Bill is an environmental scientist at the Illinois Department of Natural Resources. He wants to see temperature readings for the Fox river for the past two years; he is not interested in what surveys they were collected by, but he does want to be sure that he is comparing data of comparable quality and collection method. He goes to the main web page for the water quality database and logs in with his email address and password. Since he is set up in the system as a “data consumer” user, he is presented with a screen to set up the report he wants to see. He

selects “all surveys” as the data source, chooses all the Fox River as the target river, and chooses a data range that covers the past year. He then clicks a button labeled “Choose data points”, which presents him with screen where he can select from all of the MeasureTypes that have been taken on the Fox river in the past year, and he can choose what level of collector expertise he will accept. He selects “temperature/degrees C/reading 12 inches from stream bank” as the data point he wants to see, and chooses “trained amateur” as the minimum level of collector expertise he is willing to accept.

He clicks “Generate report” and is presented with the above data in a table format, grouped by data collection site. From here he has a number of display options, including downloading the data as a CSV-type file for importing into Excel so that he can work with the data in more detail and generate graphs he needs for an upcoming presentation. In addition, he can choose to export this particular set of data in EML format, for importing into an ecological model his group at the DNR is working with.

## **Conclusion**

The design outlined here should provide a workable basis for a sustainable, extensible, and interoperable water quality survey database. Future work on implementation should take care to follow the principles of data quality monitoring outlined here, and should make an effort to have this databased used by as wide an audience as possible. The “biodiversity commons” [5] requires the availability of easily accessible, public-domain data, and this effort can make a significant contribution in the area of ecological modeling by disseminating data collected in the state of Illinois as widely as possible.

## References

1. Chapman, Arthur D., Muñoz, Mauro E.S., and Koch, Ingrid. (2005). Environmental information. *Biodiversity Informatics* 2 24-41
2. RiverWatch. (2002). *Illinois RiverWatch Stream Monitoring Manual: Revised 5th edition*. Illinois Department of Natural Resources. Springfield, IL
3. Van House, Nancy A. (2002). Trust and epistemic communities in biodiversity data sharing. *JCDL* 2002
4. Chapman, Arthur. (2005). *Principles of Data Quality*. Global Biodiversity Information Facility.  
[http://www.gbif.org/prog/digit/data\\_quality/](http://www.gbif.org/prog/digit/data_quality/)
5. Mortiz, Thomas. (2002). Building the biodiversity commons. *D-Lib Magazine* 8

## Appendix A: Schema Documentation

**Collector** – a person who collects data for a survey.

- CollectorID (int) – primary key
- AffiliationID (int) – foreign key to **Affiliation**
- AuthPriv (varchar) – holds a privilege token, such as “administrator”, “volunteer”, etc, which is used by the front-end interface for access rights in the context of this particular **Survey**
  - Recommended user classes:
    - “administrator”: super-user, can do anything to anything; use with care
    - “survey\_creator”: able to create new **Surveys** and add **Collectors** to the system; can assign privileges to **Collectors** to give them rights to edit surveys
    - “data\_consumer”: able to view/report data from all **Surveys**
  - NOTE that AuthPriv exists here AND in **CollectorToSurvey**. This is necessary so that **Collectors** can have global privileges (assigned here) and survey-specific privileges (assigned in **CollectorToSurvey**). The privileges elaborated here are those thought to be most appropriate at the global level.
- Name (varchar) – collector's name
- Address (varchar)– collector's address
- Email (varchar) – collector's email address; also used as their login to the data collection system
- Password (varchar) – encrypted password string

**TrainingLevel** – the level of training for a **Collector**. **TrainingLevels** apply to *each MeasureType* that the collector is certified for; there's no such thing as a collector that is an “expert” in every measurement across the board.

- TrainingLevelID (int) – primary key
- CollectorID (int) – foreign key to **Collector**
- MeasureTypeID (int) – foreign key to MeasureType; tells us what particular measurement this training applies to
- Date (date) – date of training
- Rank (varchar) – level of training; something like “professional”, “amateur”, etc. If a collector does *not* have a **TrainingLevel** for a particular **MeasureType**, it is assumed that he/she is untrained for that measure type

**Affiliation** – organization above the **Survey**. Also used for **Collectors**, because one **Collector** can a) participate in more than one **Survey**, and b) a **Collector** may have an **Affiliation** with an organization that has no surveys (i.e., a University), but that we should record as the **Collector's** primary institutional affiliation.

- AffiliationID (int) – primary key
- Name (varchar) – name of organization
- Address (varchar) – contact information

**Survey** – an entity that administers **CollectionEvents**.

- SurveyID (int) – primary key
- ProtocolID (int) – foreign key to **Protocol**

- AffiliationID (int) – foreign key to **Affiliation**
- Name (varchar) – name of this survey
- Description (text) – description of what this survey does, etc

**River** – a body of water, monitored by a **Survey**

- RiverID (int) – primary key
- Name (varchar) – name of the river

**SurveyToRiver** – many-to-many mapping table for **Survey** and **River**

- RiverID (int) – foreign key to **River**
- SurveyID (int) – foreign key to **Survey**

**CollectorToSurvey** – many-to-many mapping table associating **Collectors** with **Surveys**

- CollectorID – foreign key to **Collector**
- SurveyID – foreign key to **Survey**
- AuthPriv (varchar) – holds a privilege token, such as “administrator”, “volunteer”, etc, which is used by the front-end interface for access rights in the context of this particular **Survey**
  - Recommended user classes:
    - “survey\_admin”: able to add new **MeasureTypes**, **Protocols**, and **Collectors** to their survey
    - “data\_collector”: able to input measurements to **Surveys** they are associated with; also able to view/report data from **Surveys** they are associated with
    - “survey\_data\_consumer”: able to view/report data from associated **Surveys**
    - NOTE that AuthPriv exists here AND in **Collector**. This is necessary so that **Collectors** can have global privileges (assigned in **Collector**) and survey-specific privileges (assigned here). The privileges elaborated here are those thought to be most appropriate at the survey level.

**CollectionEvent** – an instance of measurement-gathering in the field. These are run by **Surveys** and are typically periodic (i.e., monthly, annually, etc).

- EventID (int) – primary key
- SurveyID (int) – foreign key to **Survey**
- Name (varchar) – name of the event; all unique past **CollectionEvent** names should be displayed in the data entry interface, and selecting one will clone the attributes of the past event into a new one. This is to accommodate periodic data collection (which is how most data collection is done).
- DateTime (datetime)
- Description (text) – description of the event, like “Annual collection event on the Fox River at Huntley, IL”
- LatLong (varchar) – precise location of the data collection

**Protocol** – data collection protocol. Described by all of the **MeasureTypes** that must be gathered to fill the data required for the **Protocol**. Associated with its **MeasureTypes** through **ProtToMeasureType**.

- ProtocolID (int) – primary key

- Name (varchar) – short name of this protocol
- Description (text) – long and *descriptive* characterization of the protocol

**MeasureType** – kinds of measurements. Many of these taken together constitute a **Protocol**.

- MeasureTypeID (int) – primary key
- DataTypeID (int) – foreign key to **DataType**
- Name (varchar) – name of this measurement; repeated names are used to indicate multiple options for collection (i.e., pH paper v. electronic pH meter)
- Description (text) – description of this measurement
- Units (varchar) – units for the measurement, i.e. “ppm” or “mL”
- MinExpectedVal (varchar) – minimum “reasonable” value for this item; used for data validation
- MaxExpectedVal (varchar) – maximum “reasonable” value
- MinAllowedVal (varchar) – absolute minimum; i.e., you can't have <0 ppm of anything
- MaxAllowedVal (varchar) – absolute maximum; i.e., you can't have liquid water at 200C
- MeasurementMethod (text) -- detailed description of how this particular item is to be collected in the field

**DataType** – typecast of a **MeasureType**; used for data validation and output.

- DataTypeID (int) – primary key
- Type (varchar) – data cast; INT, STRING, REAL, etc.
  - Suggested **DataTypes** for the implemented system are:
    - INT
    - REAL
    - TEXT (free text input)
    - SELECT (a multiple selection interface)
    - SELECT+OTHER (a multiple selection, plus an “other” field allowing free text input)
  - Under both SELECT types, the MaxExpectedVal attribute of the **MeasureType** should be populated with a comma-separated list of possible values, for presentation in the data input interface
  - All of TEXT, SELECT, and SELECT+OTHER store text strings in **Measure**, and need to be cast back to text strings for reporting; they are differentiated in **DataType** to accommodate data input interface construction.

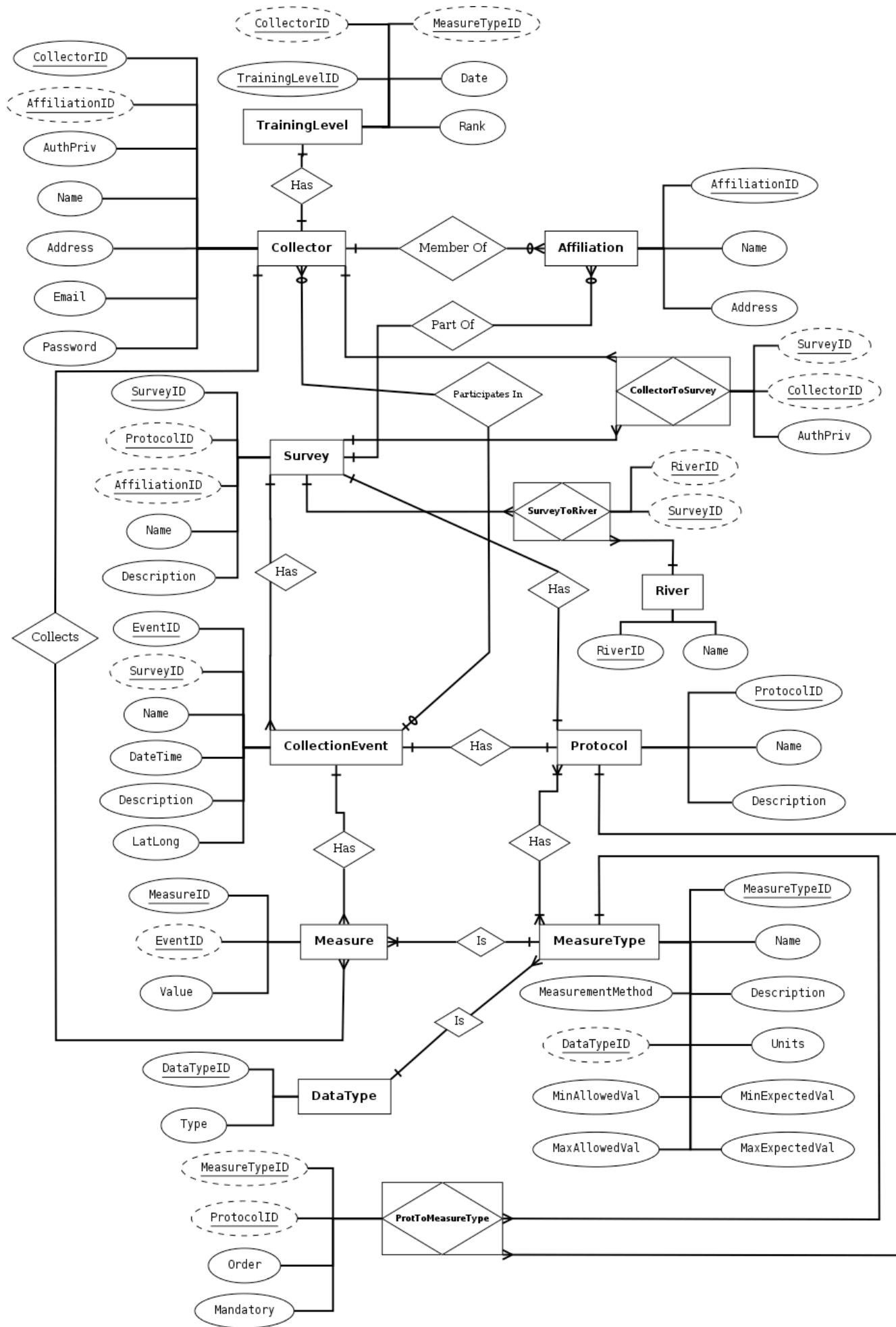
**Measure** – data points

- MeasureID (int) – primary key
- EventID (int) – foreign key to **CollectionEvent**
- Value (blob) – value of the data item; must be cast to its real type for reporting, via its **DataType**

**ProtToMeasureType** – many-to-many cross-table for associating **Protocols** with their **MeasureTypes**.

- MeasureTypeID (int) – foreign key to **MeasureType**

- ProtocolID (int) – foreign key to **Protocol**
- Order (int) – display/collection order of this particular item (0...n)
- Mandatory (boolean) – is this a required or optional data point?



## Appendix B: Previous schema documentation

