# NeuroCore TimeSeries Datablade

**Jonas Mureika and Edriss N. Merchant**
*University of Southern California Brain Project,*
*University of Southern California, Los Angeles, California*

## Introduction

To be able to store neurophysiological and behavioral data (for more information on the experiments and data being stored within NeuroCore, please see Chapter 3.1) in a usable format within the database, a new data type had to be created. In an object-relational database, this new type can be accessed and manipulated by the database just as for the database's inherent types (e.g., integer, float, varchar). Being able to directly access and manipulate this new time series data types allows users to perform queries and perform analysis on the data directly and not just on metadata that has been computed at some prior time.

## Internal Structure and Description

The NeuroCore TimeSeries Datatype is constructed by using Informix's opaque data type. The opaque data type is a fundamental data type that a user can build upon (a fundamental data type is atomic; it cannot be broken into smaller pieces, and it can serve as the building block for other data types). When one defines an opaque data type, the data type system of the database server is extended. The new opaque data type can be used in the same way as any built-in data type that the database server provides. To define the opaque data type to the database server, one must provide the following information:

- A data structure that serves as the internal storage of the opaque data type
- Support functions that allow the database server to interact with this internal structure
- Optional additional routines that can be called by other support functions or by end users to operate on the opaque data type

The NeuroCore TimeSeries Datatype is able to store data in both binned (data collected at regular time intervals) and time-stamped (data collected at irregular time intervals) formats. The general structure of the data type (see Fig. 1 for an example of one-dimensional binned data and Fig. 2 for an example of two-dimensional time-stamped data) contains a header, data record, and optional time record. The header itself stores vital information regarding the structure of the data being stored. The data record contains the actual data being stored, whereas the time record contains the collection time points if the data were collected at irregular time intervals (it is NULL otherwise). The structure of the NeuroCore TimeSeries Datatype can also be extended to the N-dimensional case (e.g., for the storage of three-dimensional coordinates recorded from a moving object).

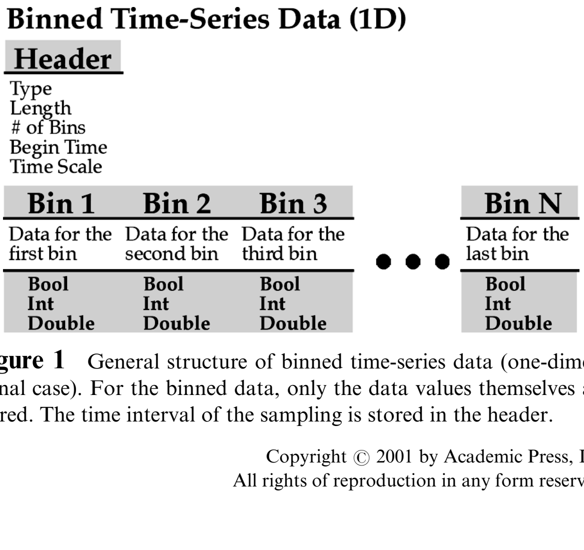

**Figure 1** General structure of binned time-series data (one-dimensional case). For the binned data, only the data values themselves are stored. The time interval of the sampling is stored in the header.

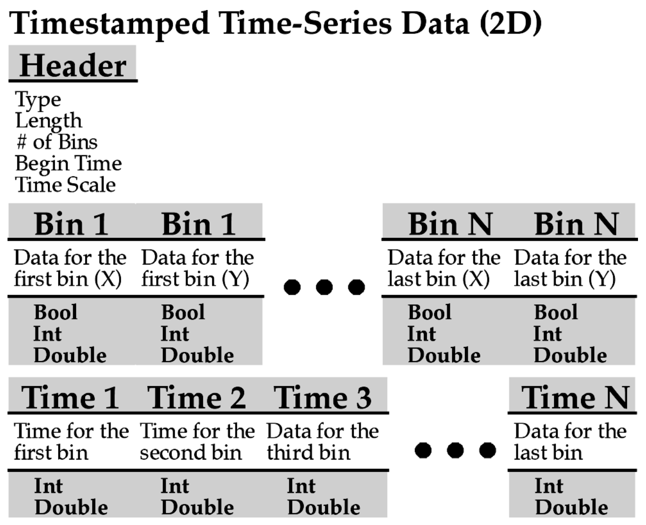

**Figure 2** Internal format of time-stamped time-series data (two-dimensional case). For the time-stamped data, the data values are stored along with a time stamp.

## Support Functions

The support functions of an opaque data type tell the database server how to interact with the internal structure of the opaque type. Because an opaque type is encapsulated, the database server has no *a priori* knowledge of its internal structure. It is the support functions, which one writes, that interact with this internal structure. The database server calls these support functions to interact with the data type. The table found on the next page summarizes the support functions one needs to define for an opaque data type.

## Additional SQL-Invoked Routines

### Operator Functions

An operator function is a user-defined function that has a corresponding operator symbol. For an operator function to operate on a newly defined opaque data type, one must write a version of the function for the new data type. When one writes a new version of an operator function, the following rules must be adhered to:

- The name of the operator function must match the name on the server; however, the name is not case sensitive -the plus() function is the same as the Plus() function.
- The operator function must handle the correct number of parameters.
- The operator function must return the correct data type, where appropriate.

Operator functions are divided into three classes: arithmetic (e.g., the plus() operator, which corresponds to +); text (e.g., the concat() operator, which corresponds to ||); relational (e.g., the lessthan() operator which corresponds to <).

### Aggregate and Built-In Functions

A built-in function is a predefined function (e.g., exp() and sqrt()) that Informix$^{TM}$ provides for use in an SQL expression. An aggregate function (e.g., min or max) returns one value for a set of queried rows. Informix supports these functions on built-in data types. To have these functions supported on an opaque data type, one must write a new version of the function following these rules:

- The name of the function must match the name on the server; however, the name is not case sensitive -the abs() function is the same as the Abs() function.
- The function must be one that can be overridden.
- The function must handle the correct number of parameters, and these parameters must be of the correct data type.
- The function must return the correct data type, where appropriate.

### End-User Routines

The server allows one to define SQL-invoked functions or procedures that the end user can use in SQL expressions and queries. These end-user routines provide additional functionality that a user might need to work with the newly defined data type. Examples of the end-user routines include the following:

- Functions that return a particular value in the opaque data type: Because the opaque type is encapsulated, an end-user function is the only way users can access fields of the internal structure.
- Casting functions: Several of the support functions serve as casting functions between basic data types that the database server uses. Additional casting functions may be written between the opaque type and other data types (built-in, complex, or opaque) of the database.
- Functions or procedures that perform common operations on the opaque data type: If an operation or task is performed often on the opaque type, an end-user routine to perform this task may be written.

## Comparing Data

The server supports the following conditions on an opaque type in the conditional clause of SQL statements:

- The IS and IS NOT operators
- The IN operator, if the equal() function has been defined

| Function | Purpose |
|---|---|
| input | Converts the opaque-type data from its external representation to its internal representation |
| output | Converts the opaque-type data from its internal representation to its external representation |
| receive | Converts the opaque-type data from its internal representation on the client computer to its internal representation on the server computer |
| send | Converts the opaque-type data from its internal representation on the server computer to its internal representation on the client computer |
| import | Performs any tasks required to process an opaque type when a bulk copy imports the opaque type in its external representation |
| export | Performs any tasks required to process an opaque type when a bulk copy exports the opaque type in its external representation |
| importbinary | Performs any tasks required to process an opaque type when a bulk copy imports the opaque type in its external representation |
| exportbinary | Performs any tasks needed to process an opaque type when a bulk copy exports the opaque type in its external representation |
| assign() | Performs any tasks required to process an opaque type before storing it to disk |
| destroy() | Performs any tasks required to process an opaque type before the database server removes a row that contains the type |
| compare() | Compares two values of opaque-type data during a sort |

- The BETWEEN operator, if the compare() function has been defined
- Relational operators, if the corresponding operator functions has been defined
- Opaque-type sorting, if the compare() function has been defined

## Discussion

The NeuroCore TimeSeries Datatype has been developed to store the complex time-series data acquired during many neurophysiological and behavioral experiments. For other neuroscience database applications, unique data types can be constructed to support the non-standard data that are an integral part of the experiments one wishes to store. For example, a database for neuro-imaging data (see Chapter 3.1 for a discussion of this type of data) must store the three-dimensional images acquired as a part of these experiments. Defining a new opaque data type enables the user to work with complex data as easily as one can work with the standard data types found in relational databases. Furthermore, by allowing a user to access and manipulate the actual data within a non-standard data type, complex queries and data-mining procedures can be developed. For more information on the NeuroCore TimeSeries Datatype, please visit the USCBP website.