Hall D Calibration Database Table Design and Interface

N. Kolev *

Department of Physics, University of Regina, Regina, SK, S4S 0A2, Canada

Abstract

An initial version of the table design for the Hall D calibration and parameters database and the interface to communicate with the database server is reported herein. The database was created and tested in MySQL 4.0.20 and the interface was compiled and tested under gcc 3.4.1. The current version does only very essential validation and error handling, so when using the interface exact observation of the rules and conventions is recommended. A simple CGI web interface was also developed in order to allow the execution of sample tasks (creating or modifying tables, reading and writing of constants) and testing and debugging.

1 Requirements

1.1 Functionality

The Hall D calibration data base must provide storage for calibration constants and parameters. As a relational database, it consists of tables with data and control variables. The calibration data are witten to separate tables for each set of each subsystem of the GlueX detector, such as the ADC pedestals of BCAL, which is an example of a channel read out type of constant, or constants concerning individual items such as effective speed of light, attenuation, position adjustments, etc. The calibration database provides interface tools (the calibration manager) for creating new tables, modifying their structure, writing the values of the calibration constants, and reading these values. Search and browse functionality can be provided as a next step.

* Corresponding author's e-mail: kolev20n@uregina.ca

1.2 Authorization Levels and "No Deletes" Policy

There are four authorization levels (or "roles") currently considered.

- The "reader" can only read values or table information (reading priviledges in MySQL). This will be the most common usage, as it concerns all reconstruction activities (including those in production).
- The "writer" is allowed to insert values to existing tables (insert priviledges for the data tables in MySQL). This will normally be done by everyone responsible for the calibration constants of the individual subsystems of the detector (for the individual calibration constants), and this can be done by hand using for example a simple web interface, and probably by an automated user for the large sets of channel read out calibration constants.
- The "composer" is allowed to create new tables (create table and insert priviledges for the control tables), normally at the initial stage of developing the database, but also for testing of new calibration schemes, and modify the structure of existing tables (physically, new tables are created in the database).
- The "administrator" has all rights and priviledges, for obvious reasons, but as has been agreed, no data will be deleted from the database (except for extraordinary cases), and the history of every addition to the database will be kept by recording the date, the officer who performs the addition, and the notes about all additions.

Consecutive values of constants are kept in the same value table as a new record, providing among other things the date of recording and run number range of applicability, and all older sets of values are still kept in the same table. The same rule applies to the description of the table structure. If it is decided that new table structure is needed for a particular set (such as three- instead of two-parameter fit), the old description is still kept and can be restored at any time. The value table with the old structure and all data in it are also kept.

1.3 Interfaces

Currently at least two interfaces are needed.

- An interface integrated in the DANA framework, which will provide constants to the framework and through it to all objects that request calibration constants, and which will request them from the database interface.
- An interface which will take care of all communications with the database, will perform validation, will keep referential integrity, and will take requests for most generally specified tasks (such as "write constants for the BCAL

ADC pedestals for runs 100 - 300") while performing all the necessary steps as risk-free as possible (locking tables, performing consistency and integrity checks, etc.)

As it was agreed that MySQL is the most appropriate application for the calibration database, the promised implementation of transactions in future releases of MySQL will greatly fascilitate some of the functionality. Currently C++ is used for both the integrated and database interfaces, but Java and especially perl database interfaces will be undoubtedly necessary. The current status assumes an XML-mediated communication between the integrated and database interface, as it provides self-descriptive and platform-independent features.

2 Database Structure

The calibration database consist of two types of tables - control and data tables. The control tables describe what is in the data tables and provide some additional information (such as constants attributes). The data tables contain the actual values of the calibration constants, along with some additional information (date and time of recording, range of runs where applicable, etc.).

2.1 Control Tables

The essential control part of the database consists of the following three tables: tblSubsystems, tblSets, and tblSetDescriptions. The first table (tblSubsystems - see Table 1) contains the names of all subsystems (BCAL, BEAM, CDC, CHERENKOV, FCAL, FDC, MAGNET, TARGET, TOF, UPV) and is created and writable by the administrator only. The second table (tblSets - see Table 2) contains all the different sets of calibration constants needed by each separate subsystem, e.g. ADCpedestals, TDCoffsets, correctionCoefficients, etc. in the third column, while the first column is dummy id and the second column is the respective subsystem. It is created by the administrator, but is writable by anyone who has "composer" privileges. A new entry in this table is added each time a new set of calibration constants is needed by the respective subsystem.

The most essential part of the control tables is tblSetDescriptions (see Table 3). It contains all sets with their context and all descriptive information about each data table. The meaning of the columns is the following:

- The column *id* is a dummy id number.

Field	Type	Null	Key	Default	Extra
dummyID	int(11)		PRI	NULL	auto_increment
subsystem	varchar(16)				

Table 1
Structure of the tblSubsystems database table.

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
subsystem	varchar(16)				
setName	varchar(32)				

Table 2
Structure of the tblSets database table.

- The column *subsystem* is the name of the subsystem (BCAL, FCAL,CDC, ADC, etc.).
- The column *setName* is the name of the calibration set (ADCpedestals, fiberItems, wireAdjustments, etc.).
- The column *context* provides a distinction if this is the default set for the respective range of runs, or a custom context, e.g. a trial new set or a set with a different number of fitting parameters, etc.
- The column *orderContext* is used internally, but also by the users with "composer" priviledges when changing the context of a table (e.g. when a trial set is being promoted to a default set for the respective runs).
- The column *kind* can have two values - "simple" and "channel",refering to whether these are a few constants with different attributes such as units or descriptions, or these are uniform large arrays of constants - normally all channels in the read out of a subsystem.
- The column *nOfElements* contains the number of elements in the respective set.
- The columns *runMin* and *runMax* contain the minimum and maximum of the run range of applicability.
- The column *datestamp* contains the date and time when this set was created.
- The column *officer* contains the name of the person that created the set.
- The column *note* contains an optional (but recommended) note about the set. This note is especially highly recommended for non-default sets for obvious reasons.

The workings of the tblSetDescriptions table is described in an example in the Examples for Calibrators section.

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
subsystem	varchar(16)				
setName	varchar(32)				
context	varchar(16)				
orderContext	int(11)				
kind	enum('channel'				
	,'simple')			channel	
nOfElements	int(11)				
runMin	int(11)				
runMax	int(11)				
datestamp	datetime				
officer	varchar(32)				
note	varchar(128)				

Table 3
Structure of the tblSetDescriptions database table.

2.2 Data Tables

The data tables are tblAttributes (see Table 4), containing all the attributes for all value tables, and the many value tables (one or more per set and context) which contain the actual values of the calibration constants. The table tblAttributes has the following columns: a dummy *id* column, the complete table name in the *tableName* column, the number of the attribute in the *attributeNumber* column (value tables with channel information have one common attribute, while tables with small but non-uniform constants such as effective speed of light or fitting parameters have a separarte attribute for each value entry). The value tables have suggestive names such as

- tbl_BCAL_ADCpedestals_default_1, or
- tbl_BCAL_gammaCorrections_3pointFit_1.

The names consist of the prefix tbl, the subsystem, the set name, the context, and the order of the context, separated by underscores. An example table is shown in Table 5 for one of the sets of constants. Apart from the actual data values, it contains columns with a dummy *id*, *runMin* and *runMax* of applicability of the values (this is the applicability of the values, and the region of runs for each separate entry must be within the limits of runs for which this table is specified in tblSetDescriptions), the *datestamp* (date and time of

creation), the name of the *officer* who created the values, and an optional but recommended *note*.

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
tableName	varchar(70)				
attributeNumber	int(11)				
attribute	varchar(32)	YES		NULL	

Table 4
Structure of the tblAttributes database table.

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
runMin	int(11)				
runMax	int(11)				
datestamp	datetime				
officer	varchar(32)				
note	varchar(128)				
value1	float	YES		NULL	
value2	float	YES		NULL	
value3	float	YES		NULL	
value4	float	YES		NULL	

Table 5
Structure of the tbl_BCAL_gammaCorrections_default_1 database table.

Examples of the tables described in the current section, with which the testing and debugging were performed, are presented in the Appendix.

3 Calibration Manager

The calibration manager is an object instantiated from the calibDBmanager class, and is the current interface for communication with the database. The methods of the calibration manager are designed in order to provide uniform output from an outside calling program or a CGI interface. Currently the calibration manager class looks like this:

```
$\#$include $<$string$>$
$\#$include $<$mysql.h$>$
```

```
using namespace std;

\#ifndef CALIBDB\_MANAGER\_H
\#define CALIBDB\_MANAGER\_H

class calibDBmanager
\{
public:
    calibDBmanager(const int task,
                        const string credentials);
    string getCalibrationCollection(const int runNumber,
                        const string what,
                        const string globalContext = ``");
    string getAllSets();
    string readValues(const int runNumber,
           const string what,
           const string globalContext = ``");
    string getSetDescription(const string setName);
    string createCalibrationSet(const string subsystem,
                    const string setName,
                    const string context,
                    const string chadd,
                    const string kind,
                    const string attributes[],
                    const string numberOfElements,
                    const string runMin,
                    const string runMax,
                    const string officer,
                    const string note);
    string writeCalibrationValues(const string tableName,
                      const int runMin,
                      const int runMax,
                      const int runMax,
                      const string officer,
                      const string note,
                      const int nOfElements,
                      const float theValues[]);
    string restoreSetAsDefault(const string someSet,
                    const string newOfficer,
                    const string newNote);
    string getSubsystems();
private:
    string readFromTable(string queryString,
              MYSQL\_RES \&queryResult);
    calibDBmanager();
```

```
    int theTask;
    string theCredentials;
    string taskResult;
\};

$\#$endif
```

The description of the methods and data members follows.

The constructor takes two arguments - task and credentials. The first one (task) is provided as an additional control mechanism for priviledges and is kept in the private data member theTask. The second one (credentials) can be "reader", "writer", or "composer", corresponding to requesting reading priviledges only on the database tables ("reader"), requesting insert priviledges to existing data tables ("writer"), and requesting create table priviledges and insert priviledges to control tables ("composer"), respectively. These three levels of priviledges exclude any deletion of tables or records, as are the requirements for the database. Only the administrator has the priviledges to delete any information from the database, and should do so only in exceptional circumstances. The credentials serve for authorization only, and not for authentication. External mechanisms (passwords, certificates) have to be provided for authentication.

The method getCalibrationCollection gets as arguments the desired run number (runNumber), the desired collection to be read (what), and the global context (globalContext - a cumulative string for all non-default sets). Valid entries for the "what" parameter are: "all" for the entire collection of constants for the desired run, "BCAL" or "FCAL" or "CDC" etc. for all sets of the respective subsystem, "BCAL:ADCpedestals" or "FCAL:TDCoffests" etc. for a specific set. It is important to include the colon in the string in the third case and not to leave any blank spaces. The globalContext argument can contain several items separated with a space, and each item must itself has a colon as a separator. An example of globalContext is the following:

"BCAL:gammaCorrections:3pointFit FCAL:TDCoffsets:316channels"

Normally the globalContext string will be used when new calibration schemes are tested, but after they are approved for production, they will be promoted to default sets, so the regular client (the reconstruction code) would rarely need to use contexts. A couple of complete examples for the method are:

getCalibrationCollection(120,"all");

getCalibrationCollection(120,"FCAL");

getCalibrationCollection(120,"BCAL:gammaCorrections",
"BCAL:gammaCorrections3pointFit");

The getCalibrationCollection method returns a space separated string containing the number of sets found and a sequence for each set containing the subsystem, the set name, the context, the order of the context, the type of constants (simple of channel), the number of elements, and the minimum and maximum run of applicability. An example output could be:

"2 FCAL ADCpedestals default 1 channel 2112 1 99999 FCAL TDCoffsets default 1 channel 316 1 99999"

The usual use of getCalibrationCollection is by the readValues method.

The method restoreSetAsDefault takes as arguments the table to be restored as the current default table for a particular set, the officer restoring it and a new note explaining the reasons for this action (note that only default sets can be restored):

restoreSetAsDefault("tbl_BCAL_default_1","New Name.","New Note.");

The method getAllSets does not take arguments and returns string with a space separated list of all existing set table names, such as:

"BCAL_ADCpedestals_default_1 BCAL_fiberItems_default_1"

The readValues method is the basic tool for retrieving values of calibration constants. It takes the same arguments as getCalibrationCollection, but returns a string containing a simple XML file, which has the self-descriptive values of all constants that were requested. A short sample XML file is:

```
$<$?xml version="1.0" encoding="UTF-8"$>$
$<$system$>$
$<$name$>$
GlueX
$<$/name$>$
$<$subsystem$>$
BCAL
$<$subsystem\_set$><$name$>$fiberItems$<$/name$>$
$<$kind$>$simple$<$/kind$>$
$<$number\_of\_items$>$8$<$/number\_of\_items$>$
$<$value$>$0.1$<$/value$>$
$<$attribute$>$fiber diamerer (cm)$<$/attribute$>$
$<$value$>$0.01$<$/value$>$
$<$attribute$>$first cladding thickness (cm)$<$/attribute$>$
$<$value$>$0.005$<$/value$>$
```

```
$<$attribute$>$second cladding thickness (cm)$<$/attribute$>$
$<$value$>$1.58$<$/value$>$
$<$attribute$>$refraction index core$<$/attribute$>$
$<$value$>$1.42$<$/value$>$
$<$attribute$>$refraction index first cladding$<$/attribute$>$
$<$value$>$1.33$<$/value$>$
$<$attribute$>$refraction index second cladding$<$/attribute$>$
$<$value$>$16$<$/value$>$
$<$attribute$>$cEff, cm/ns$<$/attribute$>$
$<$value$>$288$<$/value$>$
$<$attribute$>$attenuation length, cm$<$/attribute$>$
$<$/subsystem\_set$>$
$<$/subsystem$>$
$<$subsystem$>$
BCAL
$<$subsystem\_set$><$name$>$gammaCorrections$<$/name$>$
$<$kind$>$simple$<$/kind$>$
$<$number\_of\_items$>$6$<$/number\_of\_items$>$
$<$value$>$4$<$/value$>$
$<$attribute$>$polynomial of order$<$/attribute$>$
$<$value$>$5.6$<$/value$>$
$<$attribute$>$coef 1$<$/attribute$>$
$<$value$>$1.18$<$/value$>$
$<$attribute$>$coef 2$<$/attribute$>$
$<$value$>$-2.5$<$/value$>$
$<$attribute$>$coef 3$<$/attribute$>$
$<$value$>$0.01$<$/value$>$
$<$attribute$>$coef 4$<$/attribute$>$
$<$value$>$-3$<$/value$>$
$<$attribute$>$coef 5$<$/attribute$>$
$<$/subsystem\_set$>$
$<$/subsystem$>$
$<$/system$>$
```

This file can be used by a calling function (like the CGI web interface for display or by wget from a remote user), and in the same format can be parsed and used by the DANA-integrated calibration database interface. So in the most common usage the flow of events can be as follows:

(1) The reconstruction code requests from DANA calibration constants;
(2) It requests the constants from the integrated interface;
(3) It requests it from a web (or other) server;
(4) The server requests the constants from a calibration manager object;
(5) It, in turn, takes the values from the database and constructs the XML file, and returns it to the server;

(6) The server returns it to the integrated interface;
(7) It, in turn, parses the XML file and returns calibration constants as STL vectors to the reconstruction code.

The getSetDescription method returns the description of a single value table. The name of the table is the argument of the method.

The method createCalibrationSet creates a new table with two alternatives: create a table for a set that already exists (like "BCAL:ADCpedestals:default"), or creates a table for an entirely new set (like "CDC:positions", the assumed context if unspecified is "default"). The alternatives are chosen by the "chadd" argument, which can take values of "add" or "change". The other self-descriptive arguments have the following restrictions:

- "subsystem" must be a single word with alphanumeric characters, maximum length 16, corresponds to "subsystem" columns in the tables
- "setName" must be a single word with alphanumeric characters, maximum length 32, corresponds to "setName" columns in the tables
- "context" must be a single word with alphanumeric characters, maximum length 16, corresponds to "context" columns in the tables
- "kind" must be "simple" or "channel"
- "attributes" is a string array of the attributes for the set (it will have one element if "kind" is "channel")
- "officer" must be a string of alphanumeric characters, maximum length 32, corresponds to "officer" columns in the tables
- "note" must be a string of alphanumeric characters, maximum length 128, corresponds to "note" columns in the tables.

The method returns a message for success or the error message if unsuccessful.

The method writeCalibrationValues is a low level utility for writing a single set of constants to a specified table. A higher level method is needed if all constants from a calibration stream have to be written. An example of writeCalibrationValues is:

writeCalibrationValues("tbl_FCAL_TDCoffsets_default_1" 1, 99999, "Officers Name", "The note",428,someValues);

This assumes default context and will be the most common usage. When calibration constants and schemes are being tested, the context must be specified. In the example above the table tbl_BCAL_ADCpedestals_default_N will be used, where N is the set with the last date of modification, provided the run range matches. Designating a set as a default set is the responsibility of a calibration officer with "composer" priviledges, and for the regular reconstruction purposes no additional knowledge to that provided by the example is necessary.

The method getSubsystems returns a space separated string of the subsystems in the tblSubsystems table.

The private method readFromTable is used internally as the engine for all reading operations from the database. It takes as a first argument the SQL reading query and as a second argument a reference to a MYSQL_RES structure, which contains the result of the query. The method only uses reading access to the database. Currently an alternative sstream second argument is under consideration, as it may provide more stable and uniform communication with the different calling methods.

The private data members "theTask" and "theCredentials" are internally used for authorization purposes. The member "taskResult" is used for internal string storage and as a return value of some of the tasks performed.

4 Sample Web Interface

A simple CGI web interface was also developed to facilitate the work with the database, and as a testing and debugging tool. It is currently available only at a trial web-server for observation, testing and debugging purposes. The web interface allows reading of values, writing values, and creating/changing the structure of tables. The user provides only general information, such as run numbers, calibration sets, etc. in several steps and the interface guides this process. Only very essential validation and error handling is provided, so users are encouraged to enter only meaningful information.

5 Examples for Calibrators

As an example for using the calibration database, let us assume that a calibration set BBCAL:gammaCorrections has to be created, containing coefficients for fitting energy corrections depending on the Z entry point for gamma showers in BCAL. As a first step, a calibration manager object must be created:

calibDBmanager theManager(3,"composer");

and then the create table method called:

theResultString = theManager.createCalibrationSet("BCAL", "gammaCorrections","default","add","simple",4,1, 99999, "John Whoever", "Correcting shower dependence)";

The manager will notice that this set is new, so it will check if such a set exists, and if not, will add an entry to the tblSetDescriptions table, and will create the new table tbl_BCAL_gammaCorrections_default_1 with the specidfied values. The number 1 at the end is the context order, and serves here to distinguish tables with the same context. Then values can be written to the new table. Then let us suppose that a new fit is found to work better (or simply can be tried), now not with a second-degree polynomial, but with a fourth-degree one. Of course, this will require changes to the reconstruction code, but for the trial purpose noone wants to affect the possible current production reconstructions going on, so in addition to changes to the reconstruction code (local or private "changes"), a "private" new set of constants is needed. So this is now needed:

theResultString = theManager.createCalibrationSet("BCAL", "gammaCorrections","4pointFit","add","simple", 6,1,100,"John Whoever", "4-th order may work better");

The manager will now create a new table called:

tbl_BCAL_gammaCorrections_4pointFit_0

with the 0 at the end meaning that this is not a default table. An entry in the tblSetDescriptions will also be created. Values can be now written and used. In the integrated interaface, a global context of
"BCAL:gammaCorrections:4pointFit" must be used.

Now let us suppose that the test is successfull. The new set must be made default:

const string attributes[6] = "polynomial of order", "coef1", "coef2", "coef3", "coef4", "coef5";

theResultString = theManager.createCalibrationSet("BCAL", "gammaCorrections","default","change","simple", attributes, "6", "1", "99999", "John Whoever", "4-th order worked better");

This will create a new table tbl_BCAL_gammaCorrections_default_2 and will add the entry in the tblSetDescriptions table. Constants can be then written. The values from the trial table may be copied for the first runs (this is currently not implemented, so must be done by hand).

This set will have a later datestamp and will be used as the default set for all runs. If the new structure works well for, let us assume, runs higher than 100, then instead using 1 as runMin, 100 must be used. The reading routines will first check the run range, and only then will check which is the latest default set.

Let us suppose that at a later stage, the new fit does not prove so good, so a switch back to the second order polynomial is needed. Then this is needed:

theResultString = restoreSetAsDefault("BCAL_gammaCorrections_default_1", "John Another", "Restored to 2-point fit.");

This will add a new entry in the tblSetDescriptions only with a new datastamp and an orderContext one higher than the hishgest existing, so when a reader attempts to read calibration constants, this set will be used if no context is specified.

6 Acknowledgments

I would like to thank D. Lawrence, M. Ito, and G. Ricardi for the many fruitful discussions within the "Calibration Database Committee", R. Jones for his help on XML and other topics, and the SPARRO group at University of Regina for the useful discussions.

7 Appendix - Example Tables

id	runMin	runMax	datestamp	officer
note	value1	value2	value3	value4
1	1	99999	2006-07-21 15:29:16	NK
All defaults.	2	16.6	0.18	-3.65
2	300	480	2006-07-21 15:30:26	NK
runs 300-480 failed	2	15.6	0.18	-3.48
3	360	850	2006-07-21 15:31:15	NK
improved chi2	2	15.6	0.18	-3.49

Table 6
Contents of the tbl_BCAL_gammaCorrections_default_1 test database table.

id	subsystem	setName	context
orderContext	kind	nOfElements	runMin
runMax	datestamp	officer	note
1	FCAL	ADCpedestals	default
1	channel	2112	1
99999	2006-07-17 18:25:21	NK	Empty.
2	BCAL	fiberItems	default
1	simple	8	1
99999	2006-07-17 18:26:08	NK	Empty.
3	BCAL	gammaCorrections	default
1	simple	4	1
99999	2006-07-17 18:26:37	NK	Empty.
4	BCAL	gammaCorrections	3pointFit
0	simple	6	1
100	2006-07-17 18:27:09	NK	Empty.
5	BCAL	gammaCorrections	default
2	simple	6	1
99999	2006-07-17 18:27:47	NK	Empty.
6	FCAL	TDCoffsets	default
1	channel	316	1
99999	2006-07-17 18:28:26	NK	Empty.
7	FCAL	TDCoffsets	default
2	channel	340	210
99999	2006-07-17 18:28:45	NK	Empty.

Table 7
Contents of the tblSetDescriptions test database table.

id	tableName	attributeNumber	attribute
1	tbl_BCAL_fiberItems_default_1	1	fiber diamerer(mm)
2	tbl_BCAL_fiberItems_default_1	2	first cladding thickness (mm)
3	tbl_BCAL_fiberItems_default_1	3	second cladding thickness (mm)
4	tbl_BCAL_fiberItems_default_1	4	refraction index core
5	tbl_BCAL_fiberItems_default_1	5	refraction index first cladding
6	tbl_BCAL_fiberItems_default_1	6	refraction index second cladding
7	tbl_BCAL_fiberItems_default_1	7	cEff, cm/ns
8	tbl_BCAL_fiberItems_default_1	8	attenuation length, cm
9	tbl_BCAL_gammaCorrections_default_1	1	polynomial of order
10	tbl_BCAL_gammaCorrections_default_1	2	coef 1
11	tbl_BCAL_gammaCorrections_default_1	3	coef 2
12	tbl_BCAL_gammaCorrections_default_1	4	coef 3
13	tbl_BCAL_gammaCorrections_3pointFit_0	1	polynomial of order
14	tbl_BCAL_gammaCorrections_3pointFit_0	2	coef 1
15	tbl_BCAL_gammaCorrections_3pointFit_0	3	coef 2
16	tbl_BCAL_gammaCorrections_3pointFit_0	4	coef 3
17	tbl_BCAL_gammaCorrections_3pointFit_0	5	coef 4
18	tbl_BCAL_gammaCorrections_3pointFit_0	6	coef 5
19	tbl_BCAL_gammaCorrections_default_2	1	polynomial of order
20	tbl_BCAL_gammaCorrections_default_2	2	coef 1
21	tbl_BCAL_gammaCorrections_default_2	3	coef 2
22	tbl_BCAL_gammaCorrections_default_2	4	coef 3
23	tbl_BCAL_gammaCorrections_default_2	5	coef 4
24	tbl_BCAL_gammaCorrections_default_2	6	coef 5
26	tbl_BCAL_ADCpedestals_default_1	1	channels

Table 8
Contents of the tblAttributes test database table.