307
ANYDATASET TYPE

An ANYDATASET TYPE contains a description of a given type plus a set of data instances of that type. An ANYDATASET can be persistently stored in the database if desired, or it can be used as interface parameters to communicate self-descriptive sets of data, all of which belong to a certain type.

This chapter contains the following topics:

ANYDATASET TYPE Construction

The ANYDATASET needs to be constructed value by value, sequentially.

For each data instance (of the type of the ANYDATASET), the ADDINSTANCE function must be invoked. This adds a new data instance to the ANYDATASET. Subsequently, SET* can be called to set each value in its entirety.

The MODE of construction/access can be changed to attribute/collection element wise by making calls to PIECEWISE.

- If the type of the ANYDATASET is TYPECODE_OBJECT, individual attributes will be set with subsequent SET* calls. Likewise on access.
- If the type of the current data value is a collection type individual collection elements will be set with subsequent SET* calls. Likewise on access. This call is very similar to ANYDATA.PIECEWISE call defined for the type ANYDATA.

Note that there is no support for piece-wise construction and access of nested (not top level) attributes that are of object types or collection types.

ENDCREATE should be called to finish the construction process (before which no access calls can be made).

Summary of ANYDATASET TYPE Subprograms

This table lists the ANYDATASET Type subprograms and briefly describes them.

Table 307-1 ***ANYDATASET Type Subprograms***

Subprogram	Description
ADDINSTANCE Member Procedure	Adds a new data instance to an ANYDATASET.
BEGINCREATE Static Procedure	Creates a new ANYDATASET which can be used to create a set of data values of the given ANYTYPE.
ENDCREATE Member Procedure	Ends Creation of a ANYDATASET. Other creation functions cannot be called after this call.

Table 307-1 (Cont.) *ANYDATASET Type Subprograms*

Subprogram	Description
GET* Member Functions	Gets the current data value (which should be of appropriate type).
GETCOUNT Member Function	Gets the number of data instances in an `ANYDATASET`.
GETINSTANCE Member Function	Gets the next instance in an `ANYDATASET`.
GETTYPE Member Function	Gets the `ANYTYPE` describing the type of the data instances in an `ANYDATASET`. current data value (which should be of appropriate type).
GETTYPENAME Member Function	Gets the AnyType describing the type of the data instances in an `ANYDATASET`.
PIECEWISE Member Procedure	Sets the `MODE` of construction, access of the data value to be an attribute at a time (if the data value is of `TYPECODE_OBJECT`).
SET* Member Procedures	Sets the current data value.

ADDINSTANCE Member Procedure

This procedure adds a new data instance to an `ANYDATASET`.

Syntax

```
MEMBER PROCEDURE AddInstance(
   self          IN OUT NOCOPY ANYDATASET);
```

Parameters

Table 307-2 ADDINSTANCE Procedure Parameter

Parameter	Description
`self`	The `ANYDATASET` being constructed.

Exceptions

```
DBMS_TYPES.invalid_parameters: Invalid parameters.
DBMS_TYPES.incorrect_usage: On incorrect usage.
```

Usage Notes

The data instances have to be added sequentially. The previous data instance must be fully constructed (or set to NULL) before a new one can be added.

This call DOES NOT automatically set the mode of construction to be piece-wise. The user has to explicitly call `PIECEWISE` if a piece-wise construction of the instance is intended.

BEGINCREATE Static Procedure

This procedure creates a new ANYDATASET which can be used to create a set of data values of the given ANYTYPE.

Syntax

```
STATIC PROCEDURE BeginCreate(
   typecode     IN PLS_INTEGER,
   rtype        IN OUT NOCOPY AnyType,
   aset         OUT NOCOPY ANYDATASET);
```

Parameters

Table 307-3 BEGINCREATE Procedure Parameter

Parameter	Description
typecode	The typecode for the type of the ANYDATASET.
dtype	The type of the data values. This parameter is a must for user-defined types like TYPECODE_OBJECT, Collection typecodes, and similar others.
aset	The ANYDATASET being constructed.

Exceptions

DBMS_TYPES.invalid_parameters: dtype is invalid (not fully constructed, and like errors.)

ENDCREATE Member Procedure

This procedure ends creation of a ANYDATASET. Other creation functions cannot be called after this call.

Syntax

```
MEMBER PROCEDURE ENDCREATE(
   self              IN OUT NOCOPY ANYDATASET);
```

Parameters

Table 307-4 ENDCREATE Procedure Parameter

Parameter	Description
self	The ANYDATASET being constructed.

GET* Member Functions

These functions get the current data value (which should be of the appropriate type).

The type of the current data value depends on the MODE used for accessing it (depending on how the PIECEWISE call is invoked). If PIECEWISE has not been called, the instance is accessed in its entirety, and the type of the data value should match the type of the ANYDATASET.

If PIECEWISE has been called, the instance is accessed piece-wise. The type of the data value should match the type of the attribute (or collection element) at the current position.

Syntax

```
MEMBER FUNCTION GETBDOUBLE(
   self        IN ANYDATASET,
   dbl         OUT NOCOPY BINARY_DOUBLE)
   RETURN PLS_INTEGER;

MEMBER FUNCTION GETBFLOAT(
   self        IN ANYDATASET,
   fl          OUT NOCOPY BINARY_FLOAT)
   RETURN PLS_INTEGER;

MEMBER FUNCTION GETBFILE(
   self        IN ANYDATASET,
   b           OUT NOCOPY BFILE)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETBLOB(
   self        IN ANYDATASET,
   b           OUT NOCOPY BLOB)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETCHAR(
   self        IN ANYDATASET,
   c           OUT NOCOPY CHAR)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETCLOB(
   self        IN ANYDATASET,
   c           OUT NOCOPY CLOB)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETCOLLECTION(
   self        IN ANYDATASET,
   col         OUT NOCOPY "<collection_type>")
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETDATE(
   self        IN ANYDATASET,
   dat         OUT NOCOPY DATE)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETINTERVALDS(
   self        IN ANYDATASET,
   inv         IN OUT NOCOPY INTERVAL DAY TO SECOND)
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETINTERVALYM(
   self        IN ANYDATASET,
   inv IN OUT NOCOPY INTERVAL YEAR TO MONTH)
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETNCHAR(
   self        IN ANYDATASET,
   nc          OUT NOCOPY NCHAR)
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETNCLOB(
   self        IN ANYDATASET,
```

```
   nc          OUT NOCOPY NCLOB)
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETNUMBER(
   self        IN ANYDATASET,
   num         OUT NOCOPY NUMBER)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETNVARCHAR2(
   self        IN ANYDATASET,
   nc          OUT NOCOPY NVARCHAR2)
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETOBJECT(
   self        IN ANYDATASET,
   obj         OUT NOCOPY "<object_type>")
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETRAW(
   self        IN ANYDATASET,
   r           OUT NOCOPY RAW)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETREF(
   self        IN ANYDATASET,
   rf          OUT NOCOPY REF "<object_type>")
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETTIMESTAMP(
   self        IN ANYDATASET,
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETTIMESTAMPLTZ(
   self        IN ANYDATASET,
   ts          OUT NOCOPY TIMESTAMP WITH LOCAL TIME ZONE)
 RETURN PLS_INTEGER;

MEMBER FUNCTION GETTIMESTAMPTZ(
   self        IN ANYDATASET,
   ts          OUT NOCOPY TIMESTAMP WITH TIME ZONE)
 RETURN PLS_INTEGER,

MEMBER FUNCTION GETUROWID(
   self        IN ANYDATASET,
   rid         OUT NOCOPY UROWID)
 RETURN PLS_INTEGER

MEMBER FUNCTION GETVARCHAR(
   self        IN ANYDATASET,
   c           OUT NOCOPY VARCHAR)
   RETURN      PLS_INTEGER;

MEMBER FUNCTION GETVARCHAR2(
   self        IN ANYDATASET,
   c           OUT NOCOPY VARCHAR2)
   RETURN      PLS_INTEGER;
```

Parameters

Table 307-5 GET* Function Parameters

Parameter	Description
self	The ANYDATASET being accessed.
num	The number, and associated information., that is to be obtained.

Return Values

DBMS_TYPES.SUCCESS or DBMS_TYPES.NO_DATA

The return value is relevant only if PIECEWISE has been already called (for a collection). In such a case, DBMS_TYPES.NO_DATA signifies the end of the collection when all elements have been accessed.

Exceptions

DBMS_TYPES.INVALID_PARAMETERs: Invalid Parameters (if it is not appropriate to add a number at this point in the creation process).

DBMS_TYPES.INCORRECT_USAGE: Incorrect usage

DBMS_TYPES.TYPE_MISMATCH: When the expected type is different from the passed in type.

GETCOUNT Member Function

This function gets the number of data instances in an ANYDATASET.

Syntax

```
MEMBER FUNCTION GetCount(
   self        IN ANYDATASET)
   RETURN      PLS_INTEGER;
```

Parameter

Table 307-6 GETCOUNT Function Parameter

Parameter	Description
self	The ANYDATASET being accessed.

Return Values

The number of data instances.

GETINSTANCE Member Function

This function gets the next instance in an ANYDATASET. Only sequential access to the instances in an ANYDATASET is allowed.

After this function has been called, the GET* functions can be invoked on the ANYDATASET to access the current instance. If PIECEWISE is called before doing the GET* calls, the individual attributes (or collection elements) can be accessed.

It is an error to invoke this function before the ANYDATASET is fully created.

Syntax

```
MEMBER FUNCTION GETINSTANCE(
   self           IN OUT NOCOPY ANYDATASET)
   RETURN         PLS_INTEGER;
```

Parameters

Table 307-7 GETINSTANCE Function Parameter

Parameter	Description
self	The ANYDATASET being accessed.

Return Values

DBMS_TYPES.SUCCESS or DBMS_TYPES.NO_DATA

DBMS_TYPES.NO_DATA signifies the end of the ANYDATASET (all instances have been accessed).

Usage Notes

This function should be called even before accessing the first instance.

GETTYPE Member Function

This function gets the AnyType describing the type of the data instances in an ANYDATASET.

Syntax

```
MEMBER FUNCTION GETTYPE(
   self           IN ANYDATASET,
   typ            OUT NOCOPY AnyType)
   RETURN         PLS_INTEGER;
```

Parameters

Table 307-8 GETTYPE Function Parameter

Parameter	Description
self	The ANYDATASET.
typ	The ANYTYPE corresponding to the AnyData. May be NULL if it does not represent a user-defined function.

Return Values

The typecode corresponding to the type of the ANYDATA.

GETTYPENAME Member Function

This procedure gets the fully qualified type name for the ANYDATASET.

If the ANYDATASET is based on a built-in, this function will return NUMBER and associated information.

If it is based on a user defined type, this function will return *schema_name.type_name*. for example, SCOTT.FOO.

If it is based on a transient anonymous type, this function will return `NULL`.

Syntax

```
MEMBER FUNCTION GETTYPENAME(
   self            IN ANYDATASET)
   RETURN          VARCHAR2;
```

Parameter

Table 307-9 GETTYPENAME Function Parameter

Parameter	Description
`self`	The ANYDATASET being constructed.

Return Values

Type name of the `ANYDATASET`.

PIECEWISE Member Procedure

This procedure sets the `MODE` of construction, access of the data value to be an attribute at a time (if the data value is of `TYPECODE_OBJECT`).

It sets the `MODE` of construction, access of the data value to be a collection element at a time (if the data value is of a collection `TYPE`). Once this call has been made, subsequent `SET*` and `GET*` calls will sequentially obtain individual attributes or collection elements.

Syntax

```
MEMBER PROCEDURE PIECEWISE(
   self          IN OUT NOCOPY ANYDATASET);
```

Parameters

Table 307-10 PIECEWISE Procedure Parameter

Parameter	Description
`self`	The ANYDATASET being constructed.

Exceptions

`DBMS_TYPES.INVALID_PARAMETERS`: Invalid parameters.

`DBMS_TYPES.INCORRECT_USAGE`: On incorrect usage.

Usage Notes

The current data value must be of an object or collection type before this call can be made. There is no support for piece-wise construction or access of embedded object type attributes or nested collections.

SET* Member Procedures

This procedure sets the current data value.

The type of the current data value depends on the `MODE` with which we are constructing (depending on how we have invoked the `PIECEWISE` call). The type of the current data should be the type of the `ANYDATASET` if `PIECEWISE` has NOT been called. The type should be the type of the attribute at the current position if `PIECEWISE` has been called.

Syntax

```
MEMBER PROCEDURE SETBDOUBLE(
   self              IN OUT NOCOPY ANYDATASET,
   dbl               IN BINARY_DOUBLE,
   last_elem         IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETBFLOAT(
   self              IN OUT NOCOPY ANYDATASET,
   fl                IN BINARY_FLOAT,
   last_elem         IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETBFILE(
   self              IN OUT NOCOPY ANYDATASET,
   b                 IN BFILE,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETBLOB(
   self              IN OUT NOCOPY ANYDATASET,
   b                 IN BLOB,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETCHAR(
   self              IN OUT NOCOPY ANYDATASET,
   c                 IN CHAR,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETCLOB(
   self              IN OUT NOCOPY ANYDATASET,
   c                 IN CLOB,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETCOLLECTION(
   self              IN OUT NOCOPY ANYDATASET,
   col               IN "<collection_type>",
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETDATE(
   self              IN OUT NOCOPY ANYDATASET,
   dat               IN DATE,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETINTERVALDS(
   self              IN OUT NOCOPY ANYDATASET,
   inv               IN INTERVAL DAY TO SECOND,
   last_elem         IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETINTERVALYM(
   self              IN OUT NOCOPY ANYDATASET,
   inv               IN INTERVAL YEAR TO MONTH,
   last_elem         IN BOOLEAN DEFAULT FALSE);
```

```
MEMBER PROCEDURE SETNCHAR(
   self              IN OUT NOCOPY ANYDATASET,
   nc                IN NCHAR,
   last_elem IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETNCLOB(
  self               IN OUT NOCOPY ANYDATASET,
  nc                 IN NClob,
  last_elem          IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETNUMBER(
   self              IN OUT NOCOPY ANYDATASET,
   num               IN NUMBER,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETNVARCHAR2(
   self              IN OUT NOCOPY ANYDATASET,
   nc                IN NVarchar2,
   last_elem         IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETOBJECT(
   self              IN OUT NOCOPY ANYDATASET,
   obj               IN "<object_type>",
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETRAW(
   self              IN OUT NOCOPY ANYDATASET,
   r                 IN RAW,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETREF(
   self              IN OUT NOCOPY ANYDATASET,
   rf                IN REF "<object_type>",
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETTIMESTAMP(
   self              IN OUT NOCOPY ANYDATASET,
   ts                IN TIMESTAMP,
   last_elem IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETTIMESTAMPLTZ(
   self              IN OUT NOCOPY ANYDATASET,
   ts                IN TIMESTAMP WITH LOCAL TIME ZONE,
   last_elem         IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETTIMESTAMPTZ(
   self              IN OUT NOCOPY ANYDATASET,
   ts                IN TIMESTAMP WITH TIME ZONE,
   last_elem         IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETUROWID(
   self              IN OUT NOCOPY ANYDATASET,
   rid               IN UROWID,
   last_elem IN BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETVARCHAR(
   self              IN OUT NOCOPY ANYDATASET,
   c                 IN VARCHAR,
   last_elem BOOLEAN DEFAULT FALSE);

MEMBER PROCEDURE SETVARCHAR2(
```

```
self             IN OUT NOCOPY ANYDATASET,
c                IN VARCHAR2,
last_elem BOOLEAN DEFAULT FALSE);
```

Parameters

Table 307-11 SET* Procedure Parameters

Parameter	Description
`self`	The `ANYDATASET` being accessed.
`num`	The number, and associated information, that is to be set.
`last_elem`	Relevant only if `PIECEWISE` has been already called (for a collection). Set to TRUE if it is the last element of the collection, FALSE otherwise.

Exceptions

- `DBMS_TYPES.INVALID_PARAMETERS`: Invalid parameters (if it is not appropriate to add a number at this point in the creation process).
- `DBMS_TYPES.INCORRECT_USAGE`: Incorrect usage.
- `DBMS_TYPES.TYPE_MISMATCH`: When the expected type is different from the passed in type.