# Database

## Tables

### OBJECTS

Objects storage.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| OBJECT\_ID | N | Y | NUMBER(20) | Y | N | Object unique ID |
| PARENT\_ID | Y | Y | NUMBER(20) | N | Y | Foreign key to the parent object in  the OBJECTS table |
| OBJECT\_TYPE\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the object type in the  OBJECT\_TYPES table |
| OBJECT\_CLASS\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the object class in  the OBJECT\_TYPES table |
| PICTURE\_ID | Y | N | NUMBER(20) | N | Y | Foreign key to the picture in the  NC\_PICTURES table |
| NAME | Y | Y | VARCHAR2(200) | N | N | Name of the object |
| DESCRIPTION | Y | N | VARCHAR2(200) | N | N | Description of the object |

### OBJECT\_TYPES

Objects types definitions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| OBJECT\_TYPE\_ID | N | Y | NUMBER(20) | Y | N | Object type unique ID |
| PARENT\_ID | Y | N | NUMBER(20) | N | Y | Foreign key to the parent type in the  OBJECT\_TYPES table |
| PICTURE\_ID | Y | N | NUMBER(20) | N | Y | Foreign key to the picture in the  NC\_PICTURES table |
| NAME | N | N | VARCHAR2(200) | N | N | Object type name |
| DESCRIPTION | Y | N | VARCHAR2(200) | N | N | Description of the object type |
| PROPERTIES | Y | N | VARCHAR2(4000) | N | N | Object type properties |

### REFERENCES

Object reference parameters storage.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the referenced object  in the OBJECTS table |
| REFERENCE | N | Y | NUMBER(20) | N | Y | Foreign key to object the  OBJECTS table |
| OBJECT\_ID | N | Y | NUMBER(20) | N | Y | Order in which references will be  displayed |

### ATTRIBUTES

Attribute definitions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_ID | N | Y | NUMBER(20) | Y | N | Attribute unique ID |
| ATTR\_TYPE\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the attribute type in  the ATTR\_TYPES table |
| ATTR\_TYPE\_DEF\_ID | Y | N | NUMBER(20) | N | Y | Foreign key to the attribute  type definition in the  ATTR\_TYPE\_DEFS table |
| ATTR\_GROUP\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the attribute group in  the ATTR\_GROUPS table |
| NAME | N | N | VARCHAR2(200) | N | N | Attribute name |
| DESCRIPTION | Y | N | VARCHAR2(200) | N | N | Attribute description |
| TOOLTIP | Y | N |  | N | N | Tooltip message |
|  |  |  |  |  |  |  |

### DIRECTORY

System properties, dirrecent application settings are stored here.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| KEY | N | Y | VARCHAR2(255) | N | N | Application property name |
| VALUE | Y | N | VARCHAR2(255) | N | N | Application property value |
|  |  |  |  |  |  |  |

### PARAMS

Object parameters storage.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the attribute in the  ATTRIBUTES table |
| OBJECT\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the object in the  OBJECTS table |
| VALUE | Y | N | VARCHAR2(4000) | N | N | Attribute value |
| DATA | Y | N | CLOB | N | N | Extended attribute value |
| LIST\_VALUE\_ID | Y | N | NUMBER(20) | N | N | Foreign key to the list value in the  LIST\_VALUES table |
| DATE\_VALUE | Y | N | DATE | N | N | Date value |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

### ATTR\_OBJECT\_TYPES

'Attribute to object type' binding for object types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the attribute in the  NC\_ATTRIBUTES table |
| OBJECT\_TYPE\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the object type in the  NC\_OBJECT\_TYPES table |

### ATTR\_TYPE\_DEFS

Attribute types definitions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_TYPE\_DEF\_ID | N | Y | NUMBER(20) | Y | N | Attribute type definition unique ID |
| ATTR\_TYPE\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the attribute type in  the ATTR\_TYPES table |
| OBJECT\_TYPE\_ID | Y | N | NUMBER(20) | N | Y | Foreign key to the object type in the  OBJECT\_TYPES table |
| NAME | Y | Y | VARCHAR2(200) | N | N | Attribute type name |

### ATTR\_GROUPS

Attribute groups definitions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_GROUP\_ID | N | Y | NUMBER(20) | Y | N | Attribute group unique ID |
| NAME | N | Y | VARCHAR2(200) | N | N | Attribute group name |
| DESCRIPTION | Y | N | VARCHAR2(1000) | N | N | Attribute group description |

### ATTR\_TYPES

Abstract attribute types definitions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| ATTR\_TYPE\_ID | N | Y | NUMBER(20) | Y | N | Attribute type unique ID |
| NAME | Y | N | VARCHAR2(50) | N | N | Name or human-readable ID of the  attribute type |

|  |  |  |
| --- | --- | --- |
| **--** | **ATTR\_TYPE\_ID,** | **NAME** |
|  | 0 | ATTR\_TYPE\_TEXT |
|  | 1 | ATTR\_TYPE\_MEMO |
|  | 2 | ATTR\_TYPE\_NUMBER |
|  | 3 | ATTR\_TYPE\_DECIMAL |
|  | 4 | ATTR\_TYPE\_DATE |
|  | 5 | ATTR\_TYPE\_MASKED |
|  | 6 | ATTR\_TYPE\_URL |
|  | 7 | ATTR\_TYPE\_LIST |
|  | 9 | ATTR\_TYPE\_REFERENCE |
|  | 10 | ATTR\_TYPE\_PASSWORD |
|  | 13 | ATTR\_TYPE\_ATTACHMENT |
|  | 14 | ATTR\_TYPE\_HTML |
|  | 15 | ATTR\_TYPE\_XML |
|  | 16 | ATTR\_TYPE\_CURRENCY |
|  | 17 | ATTR\_TYPE\_MEMO |

### LIST\_VALUES

Arribute list values definitions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| LIST\_VALUE\_ID | N | Y | NUMBER(20) | Y | N | List value unique ID |
| ATTR\_TYPE\_DEF\_ID | N | Y | NUMBER(20) | N | Y | Foreign key to the attribute  type definition in the  NC\_ATTR\_TYPE\_DEFS table |
| VALUE | N | Y | VARCHAR2(200) | N | N | Value |
| SHOW\_ORDER | Y | Y | NUMBER(5) | N | N | Order in which values will be  displayed |

### PICTURES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Null Value** | **Required** | **Data Type** | **PK** | **FK** | **Description** |
| PICTURE\_ID | N | Y | NUMBER(20) | Y | N | Picture unique ID |
| NAME | Y | N | VARCHAR2(200) | N | N | Picture name |
| URL | Y | N | VARCHAR2(1000) | N | N | Picture URL |
| WIDTH | Y | N | NUMBER(20) | N | N | Picture width |
| HEIGHT | Y | N | NUMBER(20) | N | N | Picture height |

## Description of Database Schema

Unlike most databases, which store information about each entity (Location, Shelf, Port) in a separate table and use the table

columns for entity’s attribute values, We implements another approach. Our database includes two major

sets of tables: one set of table is used for storing metadata that is the definitions of object types and their attributes, another

set of tables is used for storing data itself that is objects (entities) and their attribute values. The data stored in data tables

is interpreted with use of data stored in metadata tables.

Metadata is a set of tables that defines the objects and attributes in the data tables. Using this approach provides great

flexibility for object model modification without changing database schema: if a new type of object with its own set of

attributes is introduced there is no need to create a new table, only metadata tables should be updated with the appropriate

object type/attributes definitions.

Most of the metadata can be customized while there is a small set of tables, which data is not changed after database creation.

Below is given a description of major data and metadata tables and the relations between them.

The information about all the objects is stored in 3 major tables: OBJECTS, PARAMS and REFERENCES.

Each record in OBJECTS table corresponds to a particular object.

The objects form the hierarchy; foreign key constraint applied to OBJECTS.PARENT\_ID field is used to support this.

There is also a concept of project – special object, which is used as a container for separate object trees (that is a container

for objects forest). Each object references the project it is defined within by OBJECTS.PROJECT\_ID field.

OBJECTS table stores the most generic object attributes like NAME and DESCRIPTION, while PARAMS and

REFERENCES tables are used to store the values of attributes specific to each object. A record in PARAMS

stands for one attribute value (or parameter) of particular object; a record in REFERENCES table stands for one

reference from this particular object to another one. There is a foreign key constraint (see PARAMS.OBJECT\_ID

and REFERENCES.OBJECT\_ID fields) established between each of these two tables and OBJECTS table, which

define the belonging of attribute values and references to particular object.

The information stored in PARAMS and REFERENCES tables is interpreted with use of metadata; for

storing metadata 4 major tables are used: OBJECT\_TYPES for object types, ATTRIBUTES for attributes,

ATTR\_SCHEMES for attribute schemas (see below), and a table named ATTR\_OBJECT\_TYPES, which

joins all these together. ATTR\_OBJECT\_TYPES table references OBJECT\_TYPES, ATTRIBUTES and

ATTR\_OBJECT\_TYPES by OBJECT\_TYPE\_ID, ATTR\_ID and ATTR\_SCHEMA\_ID fields accordingly (there are

correspondent foreign key constraints defined for each relationship).

Each object in the system has its object type. The object types are saved in OBJECT\_TYPES table.

Foreign key constraint supports the relationship between OBJECTS table and OBJECT\_TYPES table; the

OBJECTS.OBJECT\_TYPE\_ID field points to the related record in OBJECT\_TYPES table. The object types form

the hierarchy (similar to hierarchies of classes in programming); this hierarchy is supported by foreign key constraint (see

OBJECT\_TYPES.PARENT\_ID field). There is a root of all object types, which name is All.

Each object type has a set of attributes bound to it. The attributes themselves are stored in ATTRIBUTES table, while the

bindings of attributes to object types are saved in ATTR\_OBJECT\_TYPES table. For instance, Location object type has

attribute named ‘ZIP Code’: this means there is one row in ATTRIBUTES table, which stands for the definition of ‘ZIP

Code’ attribute, and there is one row in ATTR\_OBJECT\_TYPES table, which references Location object type record in

OBJECT\_TYPES table (by OBJECT\_TYPE\_ID field) and ‘ZIP Code’ attribute definition record in ATTRIBUTES

table (by ATTR\_ID field).

Each attribute defined in ATTRIBUTES table has its basic type. Attributes reference their basic types through

ATTR\_TYPE\_ID field, which points to the record in ATTR\_TYPES table; there is a correspondent foreign key

constraint defined between ATTRIBUTES and ATTR\_TYPES tables. There is a small set of basic attribute types,

they are: text, number, date, etc.

Basic types are non-customizable, but custom attribute types can be defined as well. There are two flavors of custom

(composite) attribute types: list attribute types and reference attribute types. The definitions of custom attribute types are

stored in ATTR\_TYPE\_DEFS table.

List attribute types are used for those attributes, which values are restricted to the list of valid ones (e.g. for ‘Status’-like

attributes). For list attribute type definition the LIST\_VALUES table is used as well. For each list type definition

(that is for each correspondent record in ATTR\_TYPE\_DEFS table) there are a number of list values (records in

LIST\_VALUES table) each referencing type definition by ATTR\_TYPE\_DEF\_ID field. Every list value has its own

unique ID (LIST\_VALUE\_ID field) and a textual representation (VALUE field).

Another flavor of composite attribute types is reference attribute types. These types are used for the attributes, which define

the reference from one object to another. The definition of reference attribute type can restrict the reference to point to

objects of particular types only. The ATTR\_TYPE\_DEFS.OBJECT\_TYPE\_ID field value points to the root of these

types’ sub-tree. For instance if the Physical Shelf type definition’s OBJECT\_TYPE\_ID field references Shelf object type,

then the attribute ‘Component’ of type Physical Shelf defined on Network Element object type allows Network Elements

to reference objects of Shelf type or one of its subtypes only.

Each attribute should belong to a particular attribute group, which is stored in ATTR\_GROUPS table. The foreign key

constraint is defined between ATTRIBUTES and ATTR\_GROUPS tables in order to support attribute groupings

(see ATTRIBUTES.ATTR\_GROUP\_ID field).

Different object types can share the same attributes. For instance both Location and Entity object types share the same ‘ZIP

Code’ attribute. There is no need to put two separate records into ATTRIBUTES table for each of these attributes;

instead there is only one record in ATTRIBUTES table and two records in ATTR\_OBJECT\_TYPES table, pointing

to the same attribute but to the different object types (Location and Entity).

If the parent object type defines a set of attributes then child object type inherits all these attributes. For instance All object

type (which is root for all other object types) has 4 system attributes defined on it: ‘Created By’, ‘Created When’, ‘Modified

By’ and ‘Modified When’. Thus all the object types in the system have those attributes (inherited from All object type). From

the database perspective it means there are only 4 records in ATTR\_OBJECT\_TYPES, which bind these attributes to

All object type.

With use of metadata the records of PARAMS and REFERENCES tables related to some particular object

(OBJECTS table) can be interpreted easily. That is each record in the PARAMS and REFERENCES tables

references the attribute definition (in ATTRIBUTES table) by ATTR\_ID field – this gives the name and (implicitly)

the type of the attribute. PARAMS.VALUE, PARAMS.LIST\_VALUE\_ID, REFERENCES.REFERENCE

fields give the value of this attribute - PARAMS.VALUE is used for storing attribute values of primitive types

(text, number), PARAMS.LIST\_VALUE\_ID references the appropriate list value from LIST\_VALUES table, and

REFERENCES.REFERENCE references another object in the system.

A special table PICTURES is used for storing picture information in the system. The pictures themselves are not stored

in the database; instead the URL of a picture file is stored in the table. OBJECT\_TYPES table and OBJECTS table

reference the PICTURES table via PICTURE\_ID field.