6.1 Item Properties

Each Item (specified in the item table) has a unique item_id and is associated with a single definition that does not change for the life of the Item* [[list exceptions to this such as ability to convert from date to pdate]]. The definition includes declaration of a dataype (item_data_type) as well as name, description and units, and Object Identifier (OID).

Each Item Data value is associated with exactly one Item definition and as such is bound to a specific datatype. The physical implementation of the OpenClinica data model (aka the database schema) stores all item data values in the item_data.value field, which is a text field of a maximum 4000 characters. Despite the fact that the DBMS treats all item data values as text, OpenClinica enforces strict data typing corresponding with the Items Datatype. Each data value must conform to the canonical format[1] for its data type. If a value with an invalid data type is inserted or updated, it must be rejected and an exception thrown.

An Item can be represented in one or more CRF Versions. Each Item has CRF version-specific properties (specified in the item_form_metadata and response_set tables) that control the allowed value domain and representation/formatting of the Item Data value in different interfaces of the system (e.g. CRF data entry input type, data import, data extract). The properties controlling these representations may change from CRF version to CRF version. The representations may be impacted by other system settings that could change over time or from user to user, such as locale settings in the case of dates, and by edit check constraints such as simple validation checks and OpenClinica Rules.

---

[1] A canonical form means that values of a particular type of resource can be described or represented in multiple ways, and one of those ways is chosen as the favored canonical form. (That form is canonized, like books that made it into the bible, and the other forms are not.) A classic example of a canonical form is paths in a hierarchical file system, where a single file can be referenced in a number of ways:

myFile.txt # in current working dir

../conf/myFile.txt # relative to the CWD

/apps/tomcat/conf/myFile.txt # absolute path using symbolic links

/u1/local/apps/tomcat-5.5.1/conf/myFile.txt # absolute path with no symlinks

The classic definition of the canonical representation of that file would be the last path. With local or relative paths you cannot globally identify the resource without contextual information. With absolute paths you can identify the resource, but cannot tell if two paths refer to the same entity.

With two or more paths converted to their canonical forms, you can do all the above, plus determine if two resources are the same or not, if that is important to your application (solve the aliasing problem).

Note that the canonical form of a resource is not a quality of that particular form itself; there can be multiple possible canonical forms for a given type like file paths (say, lexicographically first of all possible absolute paths). One form is just selected as the canonical form for a particular application reason, or maybe arbitrarily so that everyone speaks the same language. (taken from http://stackoverflow.com/questions/280107/what-does-the-term-canonical-form-or-canonical-representation-in-java-mean)