SAP NetWeaver AS ABAP Release 750, ©Copyright 2016 SAP AG. All rights reserved.

Object-Relational Mapping 10.8

Technically speaking, all objects in ABAP Objects – including persistent objects managed by the Persistence Service – exist only transiently in the internal session of an ABAP program. Conversely, AS ABAP data exists persistently in the database. The persistence of objects is achieved by object-relational mapping between database tables and the attributes of persistent classes. The mapping of existing persistent classes is defined using the Mapping Assistant of Class Builder. The support of the Mapping Assistant is stopped and no new mappings should be created.

Mapping by Business Key

When business keys are used to map objects, object-oriented programming is applied to available relational database tables in existing applications. Class Builder generated a key attribute in the persistent class for each field of the primary key of a database table. Mapping by business key involves mapping the key attributes to the primary key of an existing database table. At runtime, the Persistence Service links each managed object of the persistent class to the table entry whose key field values match the key attributes.

The key attributes of a persistent class cannot be modified. There are only GET methods, no SET methods. The key attributes form the business key of a persistent class. Thus a business key is a semantic key that is visible externally.

Example

See the persistent class CL_SPFLI_PERSISTENT. This class has a persistence mapping to columns in the database table SPFLI. The key fields CARRID and CONNID in the table SPFLI are mapped to identically named key attributes in the class. The other columns are linked to value attributes.

Mapping by Instance GUID

To map using instance GUIDs, special tables must be created that have exactly one primary key field of the type OS_GUID from ABAP Dictionary. The Persistence Service uses the key field to address the managed objects internally. The persistent class does not have a corresponding key attribute.

The content of the key column is made up of instance GUIDs. Each individual instance GUID is unique within a client. At runtime, the Persistence Service identifies each object of the persistent class using its instance GUID and assigns it to a single row in the database table accordingly. All remaining fields in the database table can then be mapped to value attributes of the persistent class.

Thus the instance GUID is a fixed length technical key that is externally invisible and that provides easy access to instances, particularly where the objects are closely interlinked.

When a business key is used to map a persistent object, a reference to it can be provided by specifying a semantic key. However, this is not possible for objects mapped using instance GUID. The reference must be obtained in a different way before the object is first accessed in a program and must then manage itself (in an internal table, for example). Mapping by instance GUID and business key can be used to simplify this process.

Mapping solely by instance GUID is best suited to objects that are purely dependent on other objects, since they do not have to be accessed explicitly in the ABAP program. These objects can be saved long-term by saving their object references persistently.

Mapping by Instance GUID and Business Key

This type of mapping is a combination of mapping by business key and mapping by instance GUID. The relevant tables must have a field of the type OS_GUID from ABAP Dictionary, as well as their application-specific primary key. Either special tables must be created or a field of type OS_GUID added to existing tables. The system generates a key attribute and thus a business key for each primary key field, just as in mapping by business key. The type OS_GUID field acts as an independent key that provides access to the object using an instance GUID. It is advisable to create this field as a unique secondary index.

Each table row can thus be uniquely identified either using the instance GUID or a business key. The Persistence Service manages the objects of persistent classes (mapped to these database tables) internally, using the instance GUID. However, it also allows them to be addressed using business keys.

In an ABAP program, for example, the business key can be used to get a reference to object of this type, which can then be used for further processing.

Single-Table Mapping

In the simplest form of mapping, exactly one database table is assigned to one persistent class. This is known as single-table mapping. At runtime, the system links one persistent object of the persistent class to exactly one entry in this table. Within an internal session, the Persistence Service provides a unique connection between the object and the database.

Multi-Table Mapping

In multi-table mapping, multiple database tables are assigned to a persistent class. Then at runtime, the system links each persistent object of this class to an entry in each of the tables involved. The key of each of these tables must have the same type.

Structure Mapping

In this case, the mapping is performed to ABAP Dictionary structures, rather than to database tables or database views. The application developer therefore has to program the type of persistent data storage in the relevant methods of the class agent.

A structure (or structures) need to be mapped, for example, if a persistent object is to contain table-like attributes. Database tables can only have flat structures, which prevents the object from being mapped to a table in this case. Even if a different form of persistent data storage is to be used (for example, files on the application server), structure mapping has to be used.