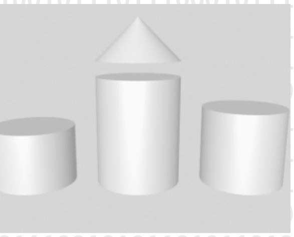


ODABA^{NG}

OR Mapping

run Software-Werkstatt GmbH
Weigandufer 45
12059 Berlin

Tel:	+49 (30) 609 853 44
e-mail:	run@run-software.com
web:	www.run-software.com

Berlin, October 2012

1 Storing ODABA data in relational databases

ODABA supports storing data in several relational databases. This is not the most efficient way of accessing data stored in ODABA, but it provides additional data access by well known SQL tools. Thus, running ODABA based on an SQL data-base might increase acceptance by customers.

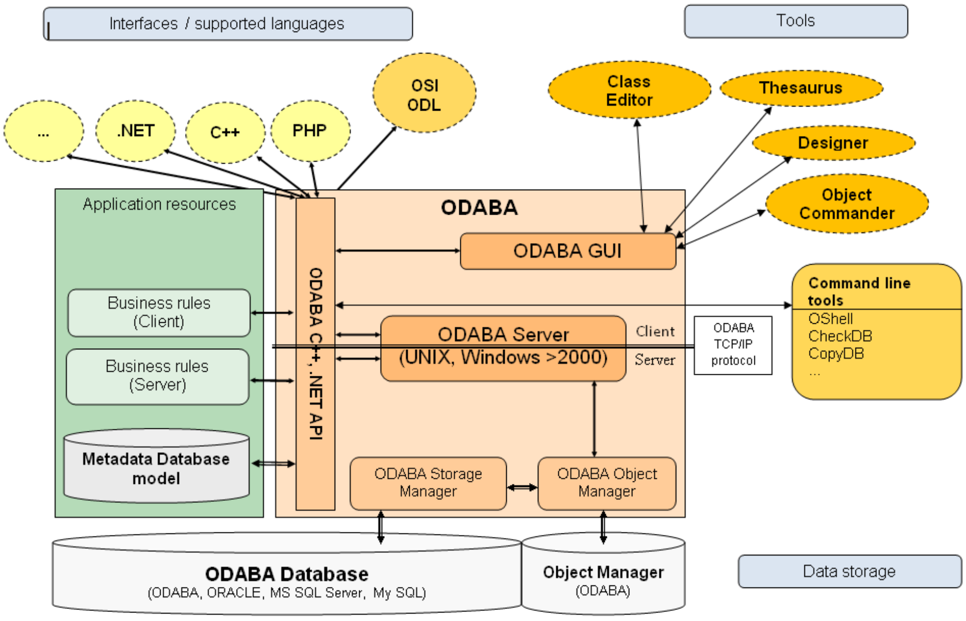
The following SQL databases have been chosen for ODABA support:

- ORACLE
- Microsoft SQL Server
- My SQL

This list might be expanded when ever required.

RDB access architecture

When running ODABA with a relational database, instances data is stored in relational tables. Optional, the administrator may decide whether to maintain m:n relationships in the RDBM or not. Thus, one may store data tables, only or data tables plus relationship tables.



In order to obtain extended ODABA features as collection events, extended instance and collection information etc. an additional database (Object Manager) is required.

Extended information as update counts for instances or collections, weak-typed or untyped collections or __IDENTITY/type mapping could hardly be handled in a relational database. Thus, an Object Manager maintains collections (relationships and references), but also update counts, locking and persistent write protection.

All services as transaction management, locking or workspace features are managed by ODABA, since SQL databases do not provide sufficient support e.g. for locking the children collection of a person. Moreover, ODABA cares about extended deletion features, maintaining inverse references and other specific object-oriented database features.

OR mapping rules

Since the information content of a relational database is a subset of the information, that can be stored in an object oriented database, mapping rules can be defined for the "relational data" in the object-oriented database.

- Instance data is stored in tables having the same name as the complex data type defined in the ODABA object model. All tables get an additional property __IDENTITY, which held the unique object identity for each instance. All relational tables are indexed by __IDENTITY.
- Complex attributes are provides as simple attributes where dots in the attribute path are replaced by double underscore (address.city --> address__city). This means that ordinary attribute names defined in the data model must not contain double underscore.
- Enumeration values are stored as numerical data. Lookup tables are not provided in the RDB automatically.
- All references and relationships, which have a collection identity (multiple and updateable references), are stored in the table as attributes with its property name proceeded by double underscore (*children* --> *__children*), which contains the identity referring to the referenced collection or instance.
- Generic attributes are considered and handled as references.
- MEMO (CLOBs - large character objects) fields and BLOBs (large binary objects) are stored in two separate tables - (_BLOB and _MEMO) - which consist of the __IDENTITY attribute and the CLOB or BLOB field.
- When a data type inherits shared from its base structure, attributes are stored in separate table for the base type using the same names as in the data model definition. The attribute referring to the base type stored the __IDENTITY value for the base instance in the referenced table. When the data type inherits exclusive from its base type, attributes of the base type become attributes of the inheriting data type.
- One to many relationships are stored as attributes with the property name of the reference or relationship. The (link) attribute contains the __IDENTITY value for the referenced instance.
- References, which always have one owner but no link field to it, will get an additional owner attribute with the name of the owning table and the attribute (e.g. Company__addresses for addresses in Company) that contains the __IDENTITY value of the owning instance.
- Many to many relationships and singular updateable relationships can be stored optionally in additional tables, which get the name according to the involved primary relationship name and the data type defining the primary relationship separated by '__' (e.g. children__Person).

The last rule is optional, since in an object model there are often more than 10 many to many relationships defined for an object type, i.e. the generated relational database might contain e.g. 100 data tables, but 1000 relationship tables in addition. Maintaining all those tables might become a performance problem. Thus, the administrator may choose when generating the relational model (table statements), whether to support many to many relationships in the relational data storage or not.

Limitations

Running ODABA with a relational database underneath includes some restrictions. The first and most important one is, that the relational data storage might be accessed by SQL tools in order to perform queries, but not in order to update the database. All update operations must pass through the Object Manager. Otherwise, the Object Manager database might become inconsistent.

Since relational databases usually do not support namespaces for tables, data model definitions running with relational data storage must not define persistent namespaces. Instead, type names should be prefixed or marked in any other way. In the model definition, you may define object types in modules or namespaces, but those must not be marked as active namespaces, i.e. type names must be unique with the dictionary.

In order to guarantee proper maintenance of inverse relationships, ODABA supports updateable relationships. In a relational database, updateable relationships behave similar as many to many relations. This means that queries against the relational database must include an additional join operation when referring to singular links.

Other limitations are of minor importance. There are several features that require specific ODABA storage. Thus, when using workspaces, all workspace data is stored in ODABA databases and is accessible via SQL only, when the workspace data has been consolidated to the root base.

Similar, long external transactions require an external ODABA transaction database and data becomes available only after committing the external transaction.

__IDENTITY values are the base for all links and instance identification and must not be changed after being created.