Mapping data model to object model

Visual Paradigm supports Object Relational Mapping (ORM) which maps data model to object model and vice versa. Visual Paradigm helps mapping entities to classes. Source files can then be generated from classes for use in software development. The mapping to object model preserves the data, but also the state, foreign/primary key mapping, difference in data type and business logic. Thus, you are not required to handle those tedious tasks during software development.

Mapping entities to classes

All entities map one-to-one to persistent classes in an object model. Classes that map with entities are represented by the stereotype .

In the above example, the *Customer* entity map one-to-one the *Customer* class as the *Customer* instance can store the customer information from the *Customer* Entity.

Mapping columns to attributes

Since all entities map one-to-one to persistent classes in an object model, columns in turn map to attributes in a one-to-one mapping. Visual Paradigm ignores all specialty columns such as computed columns and foreign key columns.

Mapping data type

Column types are automatically mapped to appropriate attribute types. The following table lists out the typical mapping between data model and object model.

Data Model	Object Model
bigint	Long
binary	Byte []
bit	Boolean
blob	Blob
varchar	String
char	Character
char (1)	Character
clob	String
date	Date
decimal	BigDecimal
double	Double
float	Float
integer	Integer
numeric	BigDecimal
real	Float
time	Time
timestamp	Timestamp

Data Model	Object Model
tinyint	Byte
smallint	Short
varbinary	Byte []

Mapping primary key

As the columns map to attributes in a one-to-one mapping, primary key columns in the entity map to attributes as a part of a class.

In the example, the primary key of entity *Product*, *ProductID* maps to an attribute *ProductID* of the class *Product*.

Mapping relationship

Relationship represents the correlation among entities. Each entity of a relationship has a role, called Phrase describing how the entity acts in it. The phrase is attached at the end of a relationship connection line. Visual Paradigm maps the phrase to role name of association in the object model.

Mapping cardinality

Cardinality refers to the number of possible instances of an entity relate to one instance of another entity. The following table shows the syntax to express the Cardinality.

Type of cardinality	Description
	Zero or one intance
	Zero or more instances
	Exactly one instance
	One or more instances

In the above example, it shows that a user contains multiple accounts.

Mapping many-to-many relationship

For each many-to-many relationship between entities, Visual Paradigm generates a link entity to form two one-to-many relationships in between. The primary keys of the two entities will automatically migrate to the link entity to form the primary/foreign keys.

In the above example, Visual Paradigm generates the link entity once a many-to-many relationship is setup between two entities. To transform the many-to-many relationship, Visual Paradigm maps the many-to-many relationship to many-to-many association.

Mapping Array Table to collection of objects

Visual Paradigm promotes the use of Array Table to allow users retrieve objects in the form of primitive array.

When Visual Paradigm transforms an array table, the array table will map to an attribute with array type modifier.

In the above example, each *Contact* may have more than one phone numbers, stored in *ContactEntry_Phone*. The array table of *ContactEntry_Phone* maps into the phone attribute with array type modifier in the *ContactEntry* class.

Typical mapping between data model and object model

Object Model	Data Model