Overview of Object-Oriented Development

The concept behind object-oriented development is that an application consists of object modules (Natural subprograms) and dialog modules (Natural programs or subprograms). The object module builds an object-level interface to a complex data structure, while the dialog module communicates with the end user and invokes methods (data actions) implemented by the object module.

Note:
For more information on object-oriented development, see *Design Methodology, Natural Construct Generation*.

Because object implementation is hidden in object-oriented development, applications:

- Are simplified
- Can use more complex data models
- Are more reliable and easier to maintain because semantic integrity is enforced within the object

An important aspect of object orientation and component-based, client/server development is the granularity of application components. Components with coarse granularity tend to contain large-scale, complex operations that provide much functionality from a single request. However, they usually do not allow the calling program to fine-tune the object’s behavior or request additional functionality.

On the other hand, finely grained components allow greater control over how their functions are performed, but they often place an additional burden on the calling program as it has to call the object multiple times in a procedural fashion. Well-conceived objects strike a balance between these approaches to optimize ease-of-use and performance.

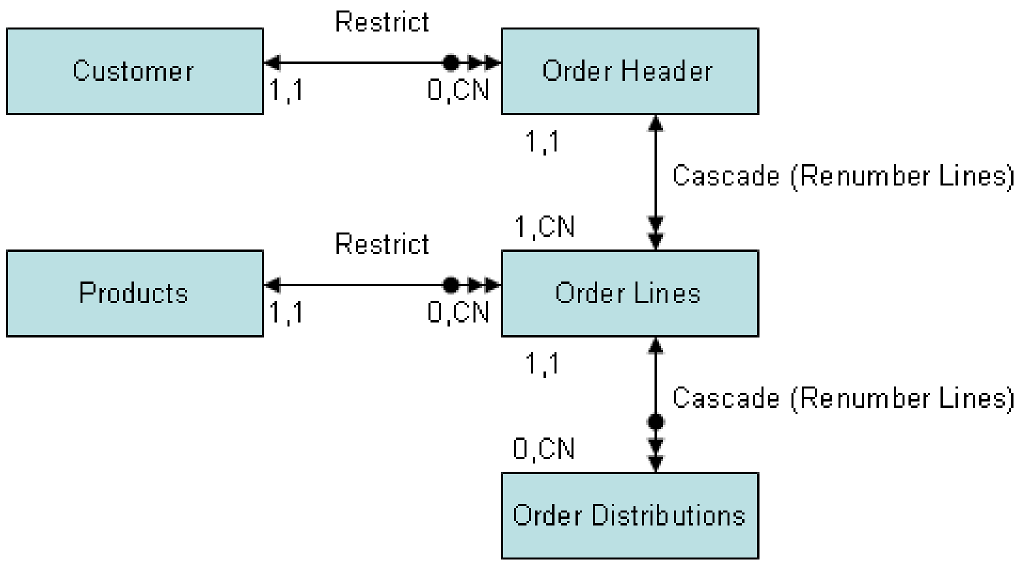
For maximum reusability, components should achieve a separation of dialog handling from business logic. This distinction allows for consistent handling of business rules, regardless of whether the rules are enforced from a GUI dialog, a browser, a mainframe screen, or through a batch process. Similarly, by distinctly separating business logic from database access, it is much easier to change the underlying database technology without affecting the business logic within applications.

This section covers the following topics:

- Define Natural Construct Objects
- Define Object Relationships in Predict

Define Natural Construct Objects

An object is a group of related entities that must be maintained within a single transaction. This section discusses some of the characteristics all objects possess and describes how to define intra-object relationships (relationships that define the bounds of an object), and inter-object relationships (relationships between two objects) in Predict. The following diagram illustrates the difference between inter-object and intra-object relationships:



1. Customer object
2. Products object
3. Order object

In this example, Customer (1) and Products (2) are related to Order (3) via inter-object relationships. The Order object is defined by two intra-object relationships between all objects.

When defining the entities within an object, you should follow certain rules. These rules are described in the following sections:

- Level 1 (Primary) File Rules
- Level 2 (Secondary) File Rules
- Level 3 (Tertiary) File Rules
- Level 4 (Quaternary) File Rules
- Primary Key Relationships

Level 1 (Primary) File Rules

All objects must have a single primary header file that contains a unique primary key field. Although you do not need to define the primary key with the unique option in the database (since the subprogram ensures its uniqueness), we recommend that you do. The primary key can be a descriptor or superdescriptor.

The primary file of a maintenance object must contain a hold field. This field is used internally by the object-maintenance subprogram to logically hold an occurrence of an object between a Get action and a corresponding Update or Delete action. The subprogram tests and updates this field to determine whether the object was updated or deleted by another user. The following table lists the valid data types for the hold field and the value assigned for each type:

Hold Field Format	Value Assigned at Updating
T	*TIMX
A10	*TIME
B8	*TIMESTMP
N7	*TIMN
A26	*TIMX (DB2 timestamp format)
If format is none of the above, it must be numeric.	Increase current value by 1.

Level 2 (Secondary) File Rules

The primary file can be related to multiple secondary files with cardinality 1:1, 1:C (1 to optionally 1), 1:N (1 to many), or 1:CN (1 to, optionally, many). Specify the maximum value for N.

Level 3 (Tertiary) File Rules

The secondary files can be related to multiple tertiary files with cardinality 1:1, 1:C, 1:N, or 1:CN. Tertiary files cannot contain multiple-valued fields (MUs) within periodic groups (PEs).

Level 4 (Quaternary) File Rules

The tertiary files can be related to multiple quaternary files with cardinality 1:1, 1:C, 1:N or 1:CN. Quaternary files cannot contain MUs or PEs.

Primary Key Relationships

Each sub-entity (any level under the primary file) must contain a key (descriptor or superdescriptor) that relates the entity to its parent entity. Because each primary key for a sub-entity (child entity) must be prefixed by the primary key of the parent entity and usually contains a suffix (used to return the sub-entities in sorted order), the primary key grows in length as you move down an object from the primary file to the secondary, tertiary, and quaternary files.

The following example shows the primary and secondary file keys for a Customer object:

File	Key
Primary	Customer-Number
Secondary	Customer-Number + Contact-Name

The following example shows the primary, secondary, and tertiary file keys for an Order object:

File	Key
Primary	Order-Number
Secondary	Order-Number + Order-Line-Number
Tertiary	Order-Number + Order-Line-Number + Distribution-Line-Number

Define Object Relationships in Predict

Use the Predict Modify File Relation panel to define object relationships in Predict. The following sections describe how to define intra-object and inter-object relationships:

- Intra-Object Relationships
- Inter-Object Relationships

Intra-Object Relationships

The following example shows the Predict Modify File Relation panel with information entered for an intra-object relationship:

```
16:44:22                 *****  P r e d I c t *****
                          - Modify File Relation -
File relation ... NCST-ORDER-HAS-LINES           Modified: 98-06-20 at 15:40
Type ...........* N  Natural Construct                 by: DEVPM
Keys ..                                                            Zoom: N

Cardinality ..* 1  : N
File 1                                             Minimum ... 1
  File ID ....* NCST-ORDER-HEADER                  Average ... 1.00
  Field ID ...* ORDER-NUMBER                       Maximum ... 1
File 2                                             Minimum ...
  File ID ....* NCST-ORDER-LINES                   Average ... 5.00
  Field ID ...* ORDER-LINE-KEY                     Maximum ... 30
Constraint attributes
  Update type .....* N  re-Number suffix
  Delete type .....* C  Cascade
  Constraint name ..
Usage ........*    (none)

Abstract     Zoom: N


EDIT:   Owner: N   Desc: N
```

The following table describes the fields on this panel and the values specified or displayed in each field for an intra-object relationship:

Field	Description
File relation	Relationship ID name. This name should not match the name of any entity within the object or any of their attributes.

Field	**Description**
Type	Relationship type. Natural Construct models process type N (Natural Construct) relationships. **Note:** For relational databases, type R relationships are also processed.
Keys	Keywords. This is an optional field.
Cardinality	
File 1	Cardinality of File 1. For an intra-object relationship, specify “1” in this field.
File-ID	Name of the file on the left side of the 1 to many (1:N) relationship.
Field-ID	Primary key for File 1. The key can be a descriptor or superdescriptor. **Note:** For relational databases, the key can be a compound key.
Minimum	Minimum number of occurrences of a record in File 1. For an intra-object relationship, specify “1” in this field.
Average	Average number of occurrences of a record in File 1. For an intra-object relationship, specify “1” in this field.
Maximum	Maximum number of occurrences of a record in File 1. For an intra-object relationship, specify “1” in this field.
File 2	Cardinality of File 2. For an intra-object relationship, specify one of the following codes: ● 1 If an occurrence of a record in File 1 must be related to exactly one occurrence of a record in File 2 and vice versa. ● C If a record in File 1 can be related to zero or one record in File 2, but a record in File 2 must be related to one record in File 1. ● N If an occurrence of a record in File 1 must be related to one or more occurrences of a record in File 2, but an occurrence of a record in File 2 must be related to exactly one occurrence of a record in File 1. ● CN If an occurrence of a record in File 1 can be related to zero or more occurrences of a record in File 2, but an occurrence of a record in File 2 must be related to exactly one occurrence of a record in File 1.
File-ID	Name of the file on the right side of the 1 to many relationship. This file is always subordinate to File 1.

Field	**Description**
Field-ID	Primary key for File 2. The key can be a descriptor or superdescriptor (or compound key). The length of this key must be greater than or equal to the length of File 1. If the length is greater, the key must be either a superdescriptor or a redefined field. The sum of the lengths of the underlying fields that prefix the superdescriptor or redefinition must also match the length of File 1.
Minimum	Minimum number of occurrences of a record in File 2 for each occurrence of a record in File 1. For an intra-object relationship, specify one of the following: ● “1” for 1:1 cardinality ● “0” for C or CN cardinality ● “N” for minimum value of N for N cardinality The generated subprogram ensures that this minimum cardinality is satisfied.
Average	Average number of occurrences of a record in File 2 for each occurrence of a record in File 1. This value is not used by Natural Construct.
Maximum	Maximum number of occurrences of a record in File 2 for each occurrence of a record in File 1. This value defines the upper bounds of the object. **Note:** When setting these maximum values, remember that generated objects must fit within the available memory.

Field	**Description**

Update type	Rules for updating the primary keys for sub-entities or testing for the existence or deletion of a sub-entity record.
	
	In the following example, you do not have to specify the line number for each order line because line numbers are determined by the position of the line within the Order object:
	
	`Object:             Order`
	`Primary File Key:   Order-Number`
	`Secondary File Key: Order-Number + Order-Line-`
	`                    Number`
	`Tertiary File Key:  Order-Number + Order-Line-`
	`                    Number + Distribution-Line-Number`
	
	In the following example, the secondary file contains contact names for the Customer object and has a key of Customer-Number + Contact-Name. Specify the suffix portion of the secondary file key:
	
	`Object:             Customer`
	`Primary File Key:   Customer-Number`
	`Secondary File Key: Customer-Number + Contact-Name`
	
	The distinction between the two types of relationships is identified by the Update constraint type. For an intra-object relationship, specify one of the following codes:
	
	• C (Cascade)
	
	Suffix portion of the sub-entity key represents data you must specify. The object-maintenance subprogram determines whether to save the sub-entity to the database (irrespective of the values of other non-primary key attributes) by the existence of a key suffix value.
	
	• L (Suffix is a line number)
	
	Suffix portion of the sub-entity key, which is a line assigned by the object-maintenance subprogram, is determined by the occurrence of the sub-entity within the object array. The existence of any non-primary key attribute within an occurrence of the sub-entity determines whether to save the sub-entity.
	
	• N (Renumber suffix)
	
	This option is similar to L, except the object-maintenance subprogram collapses empty lines (lines containing only null-valued attributes) and subsequent lines are renumbered.

Delete type	Delete constraint type. For intra-object relationships, specify “C” (Cascade), since one of the properties of an object is that all sub-entities must be deleted if the primary header file is deleted.
Constraint name	This field is not used by Natural Construct.

Inter-Object Relationships

In addition to defining relationships within objects, you can also define relationships and specify referential integrity constraints between objects. For example, you can prohibit the addition of an order for a customer who does not have a record in the Customer file (Customer object). Similarly, you can prevent the deletion of a customer who has outstanding orders in the Order file (Order object).

All referential integrity relationships must relate a foreign key of one object to a primary key of another object. The following example shows the Predict Modify File relation panel with information entered for an inter-object relationship:

```
16:54:26                 *****  P r e d I c t *****
                          - Modify File relation -
File relation ... NCST-CUSTOMER-ORDER-HEADER       Modified: 98-04-27 at 13:29
Type ...........* N  Natural Construct                   by: DEVNG
Keys ..                                                              Zoom: N

Cardinality ..* 1  : CN
File 1                                               Minimum ...
  File ID ....* NCST-CUSTOMER                        Average ... 1.00
  Field ID ...* CUSTOMER-NUMBER                      Maximum ... 1
File 2                                               Minimum ...
  File ID ....* NCST-ORDER-HEADER                    Average ... 50.00
  Field ID ...* ORDER-CUSTOMER-NUMBER                Maximum ...
Constraint attributes
  Update type .....* R  Restrict
  Delete type .....* R  Restrict
  Constraint name ..
Usage ........*    (none)

Abstract     Zoom: N


EDIT:   Owner: N   Desc: N
```

The following table describes the fields on this panel and the values specified or displayed in each field for an inter-object relationship:

Field	Description
File relation	Relationship ID name. This name should not match the name of any entity within the object or any of their attributes.
Type	Relationship type. Natural Construct models process type N (Natural Construct) relationships. **Note:** For relational databases, type R relationships are also processed.
Keys	Keywords. This is an optional field.
Cardinality	

Field	Description
File 1	Cardinality for File 1. For inter-object relationships, specify one of the following codes: ● 1 If the value of the foreign key (File 2) must match the value of the primary key (File 1). ● C If the value of the foreign key (File 2) can be blank, but if a value is specified, it must match the value of the primary key (File 1). In other words, only a non-null value in the foreign key needs to match the value of the primary key. For example, for C:N cardinality for the NCST-CUSTOMER file, it is not necessary to specify a value in the ORDER-CUSTOMER-NUMBER field in the NCST-ORDER-HEADER file. However, if a value is specified, it must have a corresponding value in the CUSTOMER-NUMBER field in the NCST-CUSTOMER file. An order does not require a customer (perhaps to signify an internal order), but if a customer is specified, it must be a valid customer. **Note:** If the foreign key is a member of a periodic group and/or a multiple-valued field, 1:CN cardinality for File 1 implies that all occurrences are required and C:CN cardinality for File 1 implies that all occurrences are optional.
File-ID	Name of the primary file for the object to which the foreign key in File 2 is related.
Field-ID	Primary key for File 1. The key can be a descriptor or superdescriptor. **Note:** For relational databases, the key can be a compound key.
Minimum	Minimum number of occurrences of a record in File 1. For an inter-object relationship, specify “1” or “0” (based on the value specified for the cardinality of File 1).
Average	Average number of occurrences of a record in File 1. For an inter-object relationship, specify “1”. This value is not used by Natural Construct during code generation.
Maximum	Maximum number of occurrences of a record in File 1. For an inter-object relationship, specify “1”.

Field	Description
File 2	Cardinality of File 2. For an intra-object relationship, specify one of the following codes: ● C If a record in File 1 can be related to a maximum of one record in File 2, but a record in File 2 must be related to one record in File 1. The value of the primary key (File 1) can be blank, but if a value is specified, it must have a corresponding value in the foreign key (File 2). For example, for 1:C cardinality for the NCST-ORDER-HEADER file, each customer specified in the CUSTOMER-NUMBER field of the NCST-CUSTOMER file can have either zero or one order specified in the ORDER-CUSTOMER-NUMBER field in the NCST-ORDER-HEADER file. If an occurrence of a record in File 1 must be related to exactly one occurrence of a record in File 2 and vice versa. ● CN If a record in File 1 does not have to be specified, but if a record is specified, it can match zero, one, or many records in File 2 (1:CN or C:CN cardinality). For example, for 1:CN cardinality for the NCST-ORDER-HEADER file, a customer specified in the CUSTOMER-NUMBER field in the NCST-CUSTOMER file can have 0, 1, or many orders specified in the ORDER-CUSTOMER-NUMBER field in the NCST-ORDER-HEADER file.
File-ID	Name of the secondary file that has a foreign key related to File 1.
Field-ID	Foreign key for File 2. The length of this key must be equal to the length of the key for File 1.
Minimum	Minimum number of occurrences of a record in File 2 for each occurrence of a record in File 1. For an inter-object relationship, specify “0” in this field.
Average	This value is not used by Natural Construct.
Maximum	For an inter-object relationship, specify the maximum number of occurrences of a record in File 2 for each occurrence of a record in File 1. **Note:** Natural Construct does not check this value to ensure it is not exceeded.
Constraint attributes	
Update type	For an inter-object relationship, specify “R” (Restrict).

Field	Description
Delete type	For an inter-object relationship, specify “R” (Restrict) to enforce the Restricted Delete rule (for information, see Support for Foreign Referential Constraints).
Constraint name	This field is not used by Natural Construct.

Support for Foreign Referential Constraints

Natural Construct supports two types of foreign referential constraints (inter-object relationships):

- Restricted Update for Insertion (RUI) rule

  When adding or updating an object entity with a foreign key related to the primary key for a related object, the action is allowed only when the value of the foreign key is either null or equal to the value of the primary key for the related object.

- Restricted Delete (RD) rule

  When deleting an object entity with a primary key that is used as a foreign key in a related object, the action is allowed only when the value of the primary key does not match any values of the foreign key for the related object.

Support for Predict Automatic Rules

Object-oriented methodology makes a distinction between data access and dialog. All data is validated and manipulated within the object-maintenance subprogram, whereas the object-maintenance dialog program communicates with the object by sending messages and parameters through the interface provided by the object parameter data area (PDA).

Since maps used by the object-maintenance dialog program do not refer to data fields directly, Natural cannot incorporate Predict automatic rules into the fields on these maps. To support object-oriented methodology, the object-maintenance subprogram incorporates the Predict automatic rules in its generated code.

Conventions for Automatic Rules

- When an error occurs, the rule should assign either the message text or number to MSG-INFO.##MSG or MSG-INFO.##MSG-NR, respectively. Optionally, the rule should also assign dynamic substitution data to MSG-INFO.##MSG-DATA(*) and then perform an ESCAPE ROUTINE. Natural Construct generates the code required to handle the error processing.

- Within the rule, specify the field name attached to the rule as “&1&”.

Note:
Only Predict verification rules with VE-STATUS = “N” (Natural Construct) are included.

Features of Automatic Rules

- You can attach a rule to any field, including a multiple-valued field (MU), a periodic group (PE), or an MU within a PE. No reference to occurrences of the array is required since the rule should not have any knowledge of what type of field it is attached to. When the rule is attached to an MU or PE, Natural Construct generates additional code to handle the array.

- A rule can declare and use external local data areas (LDAs). These LDAs can be used by another rule or by the host subprogram, since Natural Construct recognizes such usage and only generates the declaration for the LDA once.
- If a rule needs inline local data, that data must belong to a level 1 structure called #PRD-*rulename*, where *rulename* is the name of the Predict rule. This is necessary to avoid naming conflicts.
- If you want to maintain all data semantic processing within Predict and reuse any generic data processing that can be shared by multiple fields within an application, automatic rules can eliminate the need for user exit routines.