Natural für Großrechner

System-Architektur

Version 4.2.6 für Großrechner

Februar 2010

Natural

Dieses Dokument gilt für Natural für Großrechner ab Version 4.2.6 für Großrechner.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Inhaltsverzeichnis

1 System-Architektur

Diese Dokumentation beschreibt die Systemarchitektur von Natural für Mainframes und die Client-Server-Architektur des Natural Single Point of Development (SPoD). SPoD ermöglicht eine zentralisierte Entwicklung für mehrere Plattformen.

	Natural-Architektur	Hauptbestandteile der Natural-Architektur.
	Natural-SPoD-Architektur	Hauptbestandteile der Architektur von Natural Single Point of Development.

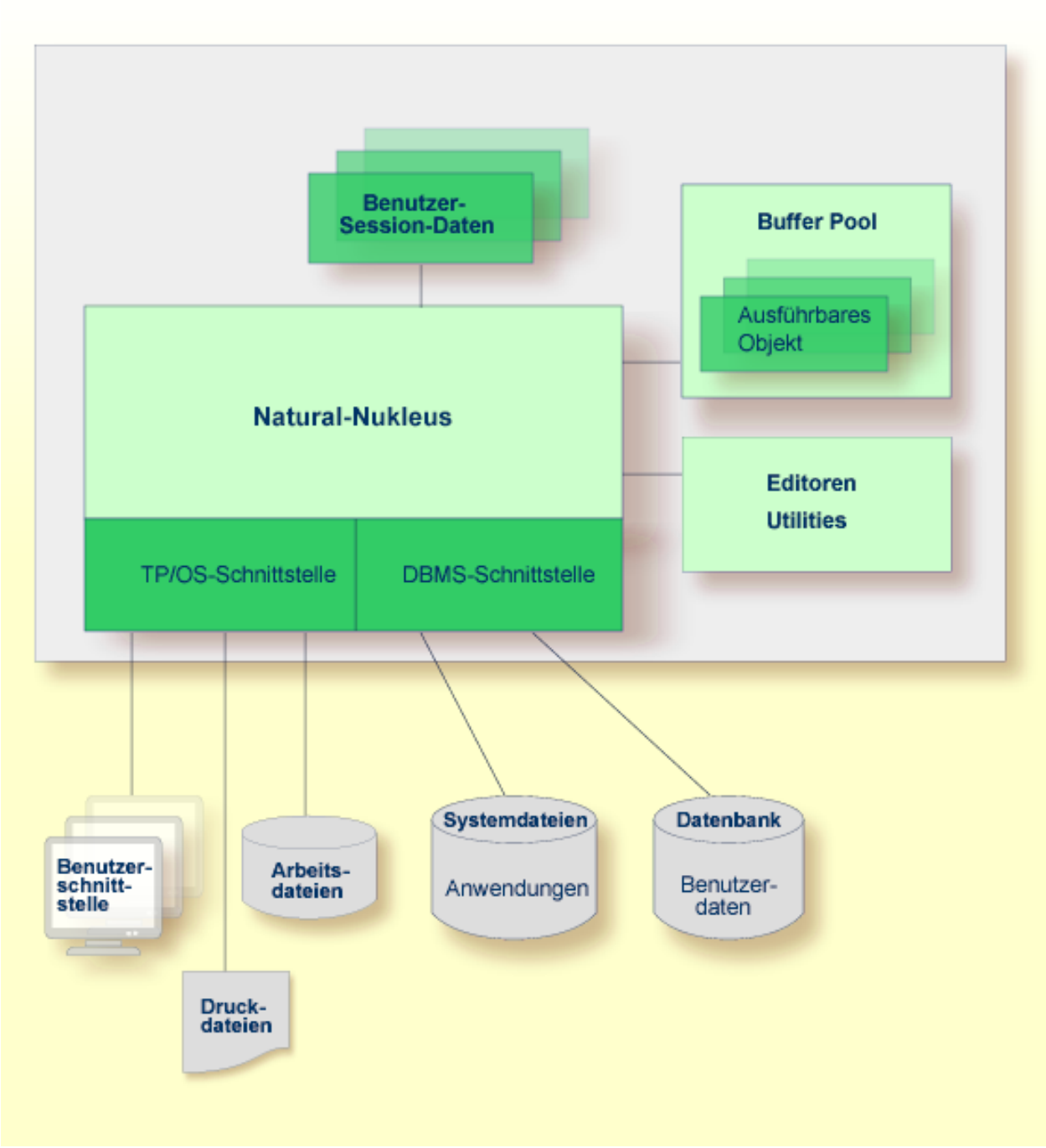
2 Natural-Architektur

Der konzeptionelle Aufbau von Natural für Großrechner basiert auf dem Prinzip einer offenen Architektur und wird durch die Verwendung von Schnittstellen realisiert. Der Informationsaustausch mit externen Komponenten erfolgt über diese Schnittstellen, und zwar sowohl während der Entwicklung einer Natural-Anwendung als auch bei der Ausführung von Programmen.

Die Natural-Systemfunktionen stehen (soweit relevant) auf allen unterstützten System-Plattformen zur Verfügung. Der Natural-Anwendungsentwickler braucht sich nicht um die komplexen Umgebungszusammenhänge des jeweiligen Systems zu kümmern (Betriebssystem, TP-System, Datenbanksystem, Benutzerumgebung).

Die Natural-Komponenten sind über spezielle systemabhängige Tabellen und Routinen in die Systemumgebung eingebettet und stellen die internen Funktionen mit den erforderlichen Ressourcen zur Verfügung.

Dieses Dokument stellt die wichtigsten Bestandteile von Natural vor und erläutert, wie diese zusammenarbeiten, um die Funktionalität einer Anwendungsentwicklungsumgebung zu realisieren.


Benutzer-
Session-Daten
Buffer Pool
Ausführbares
Objekt
Natural-Nukleus
Editoren
Utilities
TP/OS-Schnittstelle
DBMS-Schnittstelle
Benutzer-
schnitt-
stelle
Arbeits-
dateien
Systemdateien
Anwendungen
Datenbank
Benutzer-
daten
Druck-
dateien

Natural-Nukleus	Die Hauptbestandsteile des Natural-Nukleus: Compiler, Laufzeitsystem, Kommando-Interpreter und Konfiguration (Natural-Parameter).
Benutzer-Session-Daten	Temporäre Datenspeicherung in benutzerspezifischen Arbeitsbereichen.
Natural Buffer Pool	Funktionsprinzip des Natural Buffer Pool und das Laden von Objekten.
Natural-Editoren und -Utilities	Editoren und Utilities zum Erstellen und Pflegen einer Natural-Anwendung.
TP/OS-Schnittstellen	Natural-Schnittstellen zum TP-Monitor und zum Betriebssystem.
Benutzerschnittstelle	Von Natural unterstützte Benutzerschnittstellen.
Druckdateien **Arbeitsdateien**	Verwendung von Druckdateien (Print Files) und Arbeitsdateien (Work Files).
Natural-Systemdateien	Speicherung von Objektmodulen in Natural-Systemdateien.
DBMS-Schnittstelle **Datenbankzugriff**	Natural-Schnittstellen zu Datenbankverwaltungssystemen (DBMS). Datenbankzugriffe aus Natural.

3 Natural-Nukleus

Der Natural-Nukleus ist das Programm, das den Kern von Natural enthält. Der Nukleus läuft auf allen Großrechnerbetriebssystemen, z.B. z/OS, z/VSE und BS2000/OSD, und unter allen TP-Monitoren, z.B. Com-plete, CICS und *open*UTM.

Der Natural-Nukleus stellt dem Natural-Benutzer (z.B. Anwendungsentwickler oder Administrator) alle benötigten Funktionen zur Verfügung, z.B. Kommandointerpretation, Objektkompilierung und Objektausführung. Der Natural-Nukleus kann als Shared Nucleus installiert werden und kann dann zeitgleich von mehreren Benutzer genutzt werden.

Dieser Abschnitt behandelt die Hauptbestandteile des Natural-Nukleus:

Natural-Laufzeitsystem

Das Natural-Laufzeitsystem (Runtime System) funktioniert ähnlich einer virtuellen Maschine, die die zur Ausführung von Objekten in einer mit Natural erstellten Anwendung benötigte Umgebung bereitstellt. Das Natural-Laufzeitsystem interpretiert den Natural-internen Objektcode (binärer Metacode) und führt ihn aus.

Dieser Abschnitt behandelt folgende Themen:

- Objektausführung
- Object Starter und Object Executor

Objektausführung

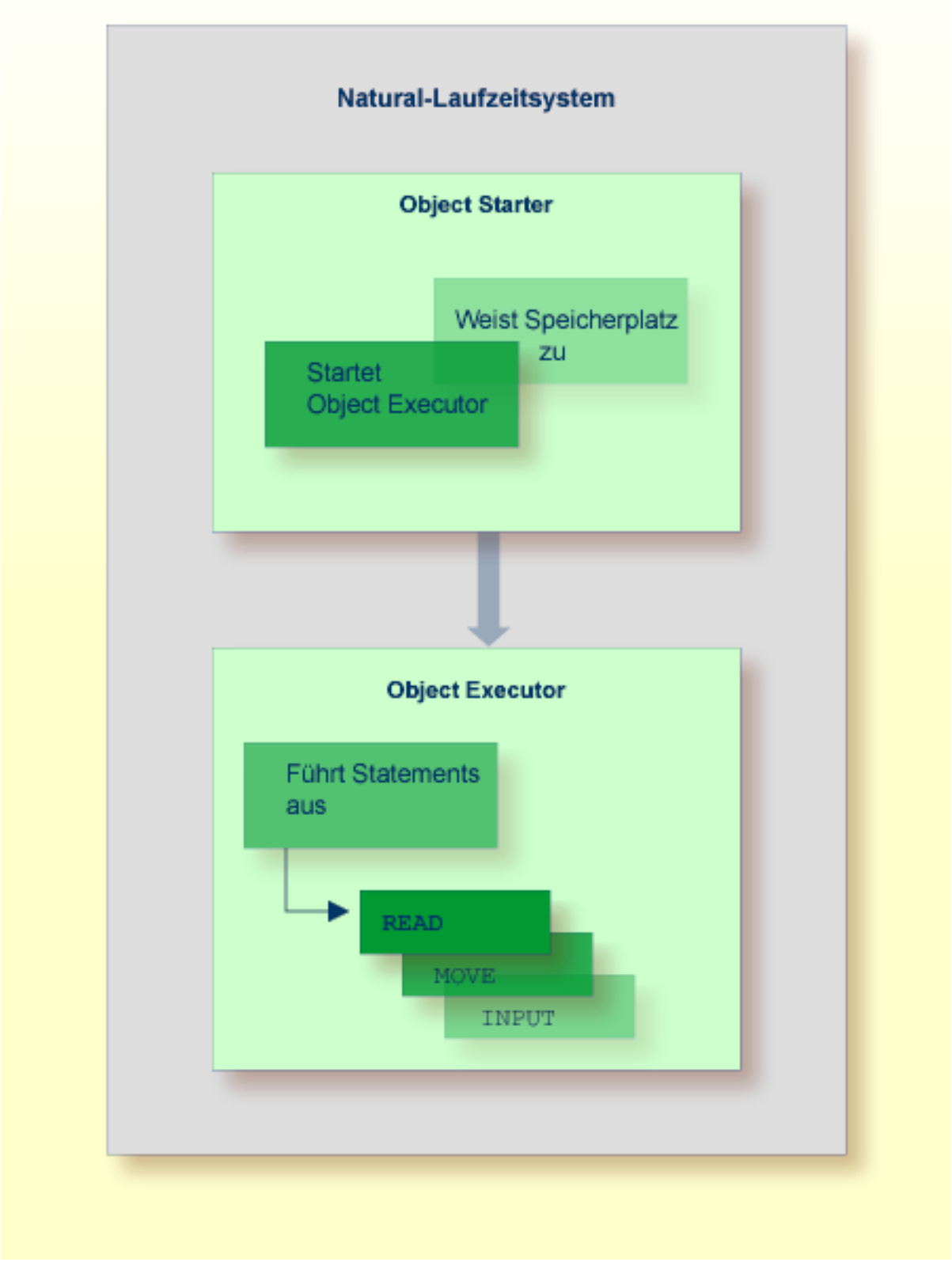
Die Ausführung des Natural-internen Objektcodes erfolgt, wenn die Ausführung eines Natural-Objekts angefordert wird.

Dies geschieht entweder direkt durch einen Benutzer oder indirekt, wenn das Objekt, das zurzeit ausgeführt wird, ein Natural-Statement absetzt, das die Ausführung eines anderen Objekts anfordert. Beispiele: Das Natural-Systemkommando `EXECUTE` bewirkt, dass das mit diesem Kommando angegebene vom Benutzer geschriebene Natural-Programm direkt ausgeführt wird. Das in einem Natural-Programm enthaltene Natural-Statement `CALLNAT` fordert die Ausführung eines Subprogramms an.

Das ***Beispiel für das Laden eines Objekts*** im Abschnitt *Natural Buffer Pool* veranschaulicht den Verarbeitungsablauf bei der Ausführung eines Natural-Programms.

Object Starter und Object Executor

In dem folgenden Diagramm ist die Ausführung eines Objekts durch das Natural-Laufzeitsystem schematisch dargestellt:


Natural-Laufzeitsystem
Object Starter
Weist Speicherplatz zu
Startet Object Executor
Object Executor
Führt Statements aus
READ
MOVE
INPUT

Das Natural-Laufzeitsystem hat zwei Hauptbestandteile: Den Object Starter und den Object Executor.

Der Object Starter ermittelt das auszuführende Objekt im **Natural Buffer Pool** und weist Speicherplatz für die vom Objekt als **Benutzer-Session-Daten** verarbeiteten Daten zu. Schließlich übergibt er die Kontrolle an den Object Executor.

Der Object Executor interpretiert die in dem Objekt enthaltenen Natural-Statements und führt sie der Reihe nach aus. In dem obigen Beispiel verarbeitet der Object Executor zunächst das `READ`-Statement, indem er einen Datenbankaufruf absetzt, die angeforderten Datensätze abruft und dann mit dem `MOVE`-Statement weitermacht.

Verwandte Themen:

- ***Benutzer-Session-Daten***
- ***Natural Buffer Pool***
- ***Beispiel für das Laden eines Objekts*** *- Natural Buffer Pool*

Natural-Compiler

Der Natural-Compiler erzeugt die ausführbare Form des Natural-Sourcecode. Beim Natural-Sourcecode handelt es sich um einen menschenlesbaren Programmcode, der im Wesentlichen aus einer Abfolge von Natural-Statements besteht. (Informationen zu den Natural-Statements und Hinweise zur Programmierung finden Sie in der *Statements*-Dokumentation bzw. im *Leitfaden zur Programmierung*.)

Der Natural-Compiler liest den Sourcecode aus dem Arbeitsbereich. Dieser Sourcecode ist Teil der Benutzer-Session-Daten. Er wird mit dem Natural-Systemkommando `READ` oder `EDIT` in den Arbeitsbereich geladen. Der Compiler prüft die Syntax des Sourcecode auf Korrektheit und erzeugt dann den Natural-internen Objektcode, der vom **Natural-Laufzeitsystem** interpretiert und ausgeführt wird.

Mit Hilfe des entsprechenden Natural-Systemkommandos kann der Benutzer den Sourcecode entweder kompilieren und ausführen oder kompilieren und speichern. Der folgende Abschnitt behandelt die verschiedenen Objektmodularten und die zur Objektkompilierung und –ausführung zur Verfügung stehenden Natural-Systemkommandos:

- Katalogisiertes Objekt
- Source-Objekt
- Kommandos zum Kompilieren

- Beispiel für eine Kompilierung

Katalogisiertes Objekt

Ein katalogisiertes Objekt ist die ausführbare (kompilierte) Form eines Natural-Objekts. Es wird vom Natural-Compiler erzeugt und als Objektmodul in einer Natural-Systemdatei gespeichert. Unter dem Katalogisieren eines Objekts versteht man das Kompilieren des Sourcecodes und das Erstellen eines katalogisierten Objekts. Erzeugt wird ein katalogisiertes Objekt mit Hilfe des Natural-Systemkommandos `CATALOG` oder `STOW`.

Zur Ausführungszeit wird das katalogisierte Objekt in den Natural Buffer Pool geladen und vom Natural-Laufzeitsystem ausgeführt. Natural-Objekte können nur ausgeführt werden oder sich gegenseitig referenzieren, wenn sie als katalogisierte Objekte in einer der Natural-Systemdateien gespeichert wurden.

Ein katalogisiertes Objekt kann nicht geändert oder dekompiliert werden.

Source-Objekt

Ein Source-Objekt (oder ein gespeichertes Objekt) enthält die menschenlesbare Form des Natural-Sourcecode. Der Sourcecode wird mit Hilfe des Natural-Systemkommandos `SAVE` oder `STOW` als Source-Objekt in einer der Natural-Systemdateien gespeichert.

Um den in einem Source-Objekt enthaltenen Sourcecode auszuführen, müssen Sie den Sourcecode kompilieren, um generierten Objektcode zu erzeugen, der vom Natural-Laufzeitsystem interpretiert und ausgeführt werden kann.

Kommandos zum Kompilieren

Natural bietet verschieden Systemkommandos zum Kompilieren des Sourcecodes. Je nach verwendetem Systemkommando wird beim Kompilieren noch eine der folgenden Aktionen ausgeführt:

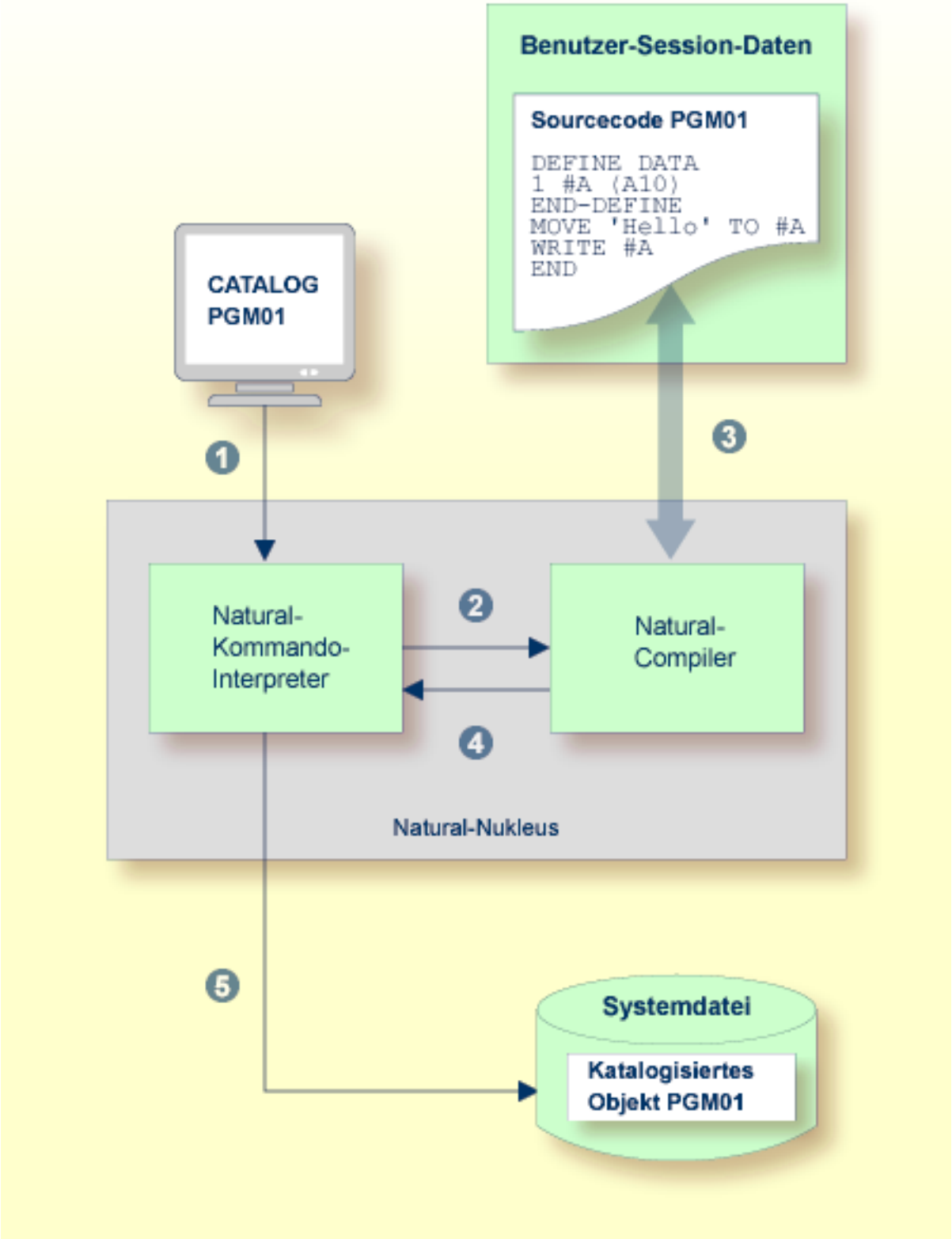
Aktion	Systemkommando
Sourcecode kompilieren. Syntaxprüfung durchführen und Objektcode generieren. Der generierte Objektcode wird nicht als Objektmodul in einer Natural-Systemdatei gespeichert.	`CHECK`
Sourcecode kompilieren. Nach erfolgreicher Kompilierung den generierten Objektcode als **katalogisiertes Objekt** in einer Natural-Systemdatei speichern.	`CATALOG`
Sourcecode kompilieren. Nach erfolgreicher Kompilierung den generierten Objektcode als **katalogisiertes Objekt** in einer Natural-Systemdatei speichern. Außerdem den ursprünglichen Sourcecode als separates **Source-Objekt** in einer Natural-Systemdatei speichern.	`STOW`
Sourcecode eines Natural-Objekts des Typs Programm kompilieren. Nach erfolgreicher Kompilierung den generierten Objektcode sofort ausführen. Der generierte Objektcode wird nicht als Objektmodul in einer Natural-Systemdatei gespeichert.	`RUN`

Verwandte Themen:

- *Systemkommandos*-Dokumentation
- *Objekttypen - Leitfaden zur Programmierung*

Beispiel für eine Kompilierung

Das folgende Diagramm veranschaulicht den Verarbeitungsablauf beim Kompilieren des Sourcecodes mit Hilfe des Systemkommandos `CATALOG`:


Benutzer-Session-Daten
Sourcecode PGM01
DEFINE DATA
1 #A (A10)
END-DEFINE
MOVE 'Hello' TO #A
WRITE #A
END
CATALOG
PGM01
1
2
3
4
5
Natural-
Kommando-
Interpreter
Natural-
Compiler
Natural-Nukleus
Systemdatei
Katalogisiertes
Objekt PGM01

Legende

1. Der Benutzer setzt das Natural-Systemkommando `CATALOG PGM01` ab, mit dem er die Kompilierung und Speicherung des zurzeit im Arbeitsbereich befindlichen Objektcodes als **katalogisiertes Objekt** mit dem Namen `PGM01` anfordert.
2. Der Natural-Kommando-Interpreter interpretiert das Kommando `CATALOG` und leitet die Anforderung an den Natural-Compiler weiter.
3. Der Natural-Compiler liest die zurzeit im Arbeitsbereich befindlichen Natural-Statements, prüft die Syntax auf Korrektheit und erzeugt dann den Objektcode.
4. Der Natural-Compiler gibt die Kontrolle an den Natural-Kommando-Interpreter zurück.
5. Der Natural-Kommando-Interpreter speichert den generierten Objektcode als **katalogisiertes Objekt** unter dem Namen `PGM01` in der aktuellen Natural-Systemdatei.

Natural-Kommando-Interpreter

Der Natural-Kommando-Interpreter prüft das in einer der Natural-Eingabeaufforderungszeilen eingegebene Kommando und führt es aus.

Wenn das Add-On-Produkt Natural Security installiert ist, kann der Administrator die Ausführung bestimmter Kommandos auf einzelne Benutzer oder eine Benutzergruppe beschränken.

Verwandte Themen:

- *Systemkommandos*-Dokumentation
- *Natural Security*-Dokumentation

Konfiguration

Die Konfiguration einer Natural-Systemumgebung wird mit Natural-Parametern verwaltet.

Diese Parameter dienen zur Standardisierung und Automatisierung der Entwicklungs- und Produktionsvorgänge und zur Anpassung der standardmäßig vorhandenen Einstellungen an die Bedürfnisse einzelner Benutzer. Mittels eines Natural-Parameters können zum Beispiel die Standardeinstellungen für die Report-Erstellung vorgenommen, die Größe eines Reports oder die Größe des erforderlichen Speicherplatzes, z.B. des Arbeitsbereichs eines Editors, festgelegt werden.

Die meisten Eigenschaften einer Natural-Umgebung sind schon von der Software AG vor der Produktauslieferung festgelegt worden. Ihr Natural-Administrator kann verschiedene Standardvorgaben konfigurieren, die für alle Benutzer in Ihrem Unternehmen gültig sind. Jeder Benutzer kann seinerseits die Einstellungen an seine Bedürfnisse anpassen, indem er die Standard-Umge-

bungseinstellungen mittels eines dynamischen Profilparameters oder eines Session-Parameters überschreibt.

Der folgende Abschnitt behandelt folgende Themen:

- Profilparameter
- Session-Parameter
- Parametrisierungsebenen

Profilparameter

Profilparameter können statisch oder dynamisch angegeben werden.

Statische Parameter werden anlässlich der Natural-Installation im Natural-Parametermodul `NATPARM` angegeben. Diese Angaben dienen als Standardvorgaben für jede Natural-Session.

Verwandte Themen:

- *Profilparameter - Parameter-Referenz*-Dokumentation
- *Profile Parameter Usage - Operations*-Dokumentation
- *Using a Natural Parameter Module - Operations*-Dokumentation
- *SYSPARM Utility - Utilities*-Dokumentation

Session-Parameter

Session-Parameter können in einer aktiven Natural-Session und/oder in einem Natural-Objekt angegeben werden. Hauptzweck der Session-Parameter ist die Steuerung der Ausführung von Natural-Programmen.

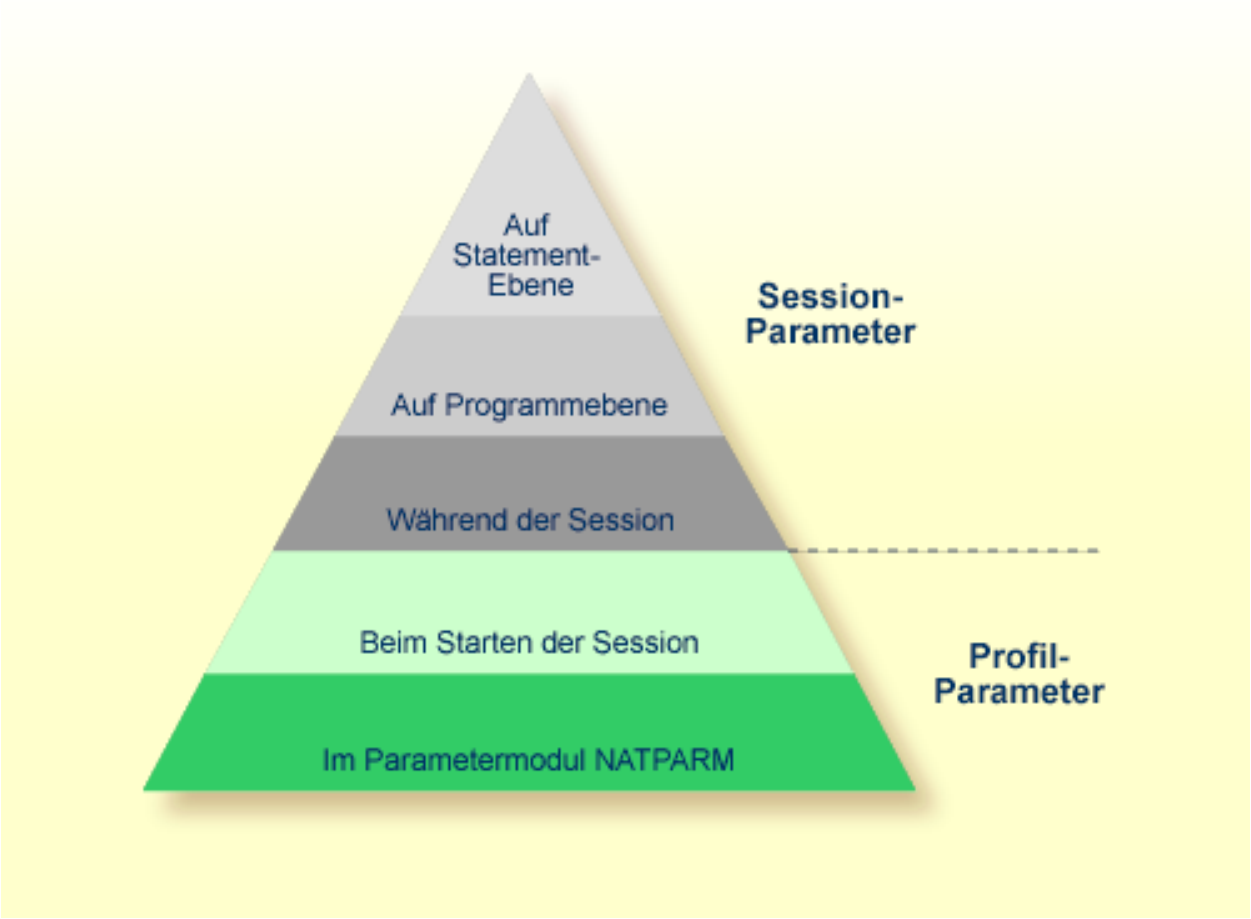
Verwandtes Thema:

- *Session-Parameter - Parameter Referenz*-Dokumentation

Parametrisierungsebenen

Die Ebenen, auf denen Natural parametrisiert werden kann, sind hierarchisch strukturiert. Ein auf einer höheren Ebene gesetzter Parameterwert hat Vorrang vor einem auf einer niedrigeren Ebene gesetzten Wert. Wenn zum Beispiel ein Parameter dynamisch angegeben wird, hat der neue Parameterwert Vorrang vor der entsprechenden statischen Parameterangabe, die im Natural-Parametermodul `NATPARM` vorhanden ist.

Das folgende Diagramm veranschaulicht, wann ein Parameter gesetzt werden kann und zeigt die Natural-Parameterhierarchie mit der niedrigsten Ebene an der Basis und der höchsten Ebene an der Spitze der Pyramide:



Verwandtes Thema:

- *Natural Parameter Hierarchy - Operations*-Dokumentation

4 Benutzer-Session-Daten

Jeder Natural-Benutzer-Session (Online oder Batch) wird ein separater Arbeitsbereich zugewiesen, der die Daten dieser Session enthält. Die Daten einer Natural-Benutzer-Session werden für die Dauer der Online- oder Batch-Session gespeichert.

Der Arbeitsbereich umfasst mehrere Natural-Buffer. Ein Buffer ist ein Zwischenspeicher, in dem ein bestimmter Datenblock, z. B. in einem Natural-Programm referenzierte Variablendaten, bereit gestellt werden.

Wie in der obigen Abbildung dargestellt, kann die Größe eines Buffers mit einem Natural-Parameter angegeben werden. Eine Ausnahme bilden die internen oder variablen Natural-Buffer. Beispielsweise wird mit dem Natural-Profilparameter `ESIZE` die Größe des Speichers festgelegt, der einem Natural-Editor zum Editieren des Sourcecodes zugewiesen ist. Der Profilparameter `DATSIZE` bestimmt zur Ausführungszeit die Größe des Speichers zur Aufnahme lokaler Daten zu einem Natural-Programm (und zu untergeordneten Objekten, die von dem Programm aufgerufen werden).

Natural bietet auch statistische Informationen zur Nutzung der der aktuellen Session zugewiesenen Buffer, zu ihrer Größe und zum tatsächlich genutzten Zwischenspeicherplatz; siehe die entsprechenden Querverweise unter *Verwandte Themen* weiter unten.

Verwandte Themen:

- *Parameter-Referenz*-Dokumentation
- *Buffer Usage Statistics - BUS*, *General SYSTP Functions*, *SYSTP Utility*, *Utilities*-Dokumentation
- *BUS - Systemkommandos*-Dokumentation

5 Natural Buffer Pool

Der Natural Buffer Pool mit gemeinsam genutztem Speicher dient als Zwischenspeicher, in den Natural **katalogisierte Objekte** (z.B. Programme) aus einer Natural-Systemdatei zur anschließenden Ausführung durch das Natural-Laufzeitsystem und/oder zur Objektkompilierung durch den Natural-Compiler lädt.

Da Natural ablauf-invarianten Natural-Objektcode generiert, kann eine einzige Kopie eines Naturalobjekts von mehr als einem Benutzer zeitgleich ausgeführt werden. Dazu wird jedes Objekt nur einmal von einer der Natural-Systemdateien in den Natural Buffer Pool geladen, anstatt von jedem Benutzer, der das Objekt anfordert, geladen zu werden.

Dieser Abschnitt behandelt grundlegende Gesichtspunkte für den Betrieb des Buffer Pools und veranschaulicht den Objektladevorgang:

Objekt in den Buffer Pool laden

Das Laden des Objekts in den Buffer Pool wird ausgelöst, wenn ein ausführbares Natural-Objekt zur Ausführung angefordert wird.

Wird die Ausführung eines Objekts angefordert, dann wird das entsprechende **katalogisierte Objekt** von der **Natural-Systemdatei**, in der es gespeichert ist, gelesen und in den Buffer Pool geladen, in dem es durch das **Natural-Laufzeitsystem** ausgeführt werden kann.

Neben ausführbaren Natural-Objekten werden auch nicht ausführbare Objekte des Objekttyps Data Area (Local, Global, Parameter) in den Buffer Pool geladen, wenn ein Natural-Objekt, das eine solche Data Area referenziert, katalogisiert oder neu katalogisiert (kompiliert) wird.

Verwandtes Thema:

- ***Objektausführung*** *- Natural-Laufzeitsystem, Natural-Nukleus*

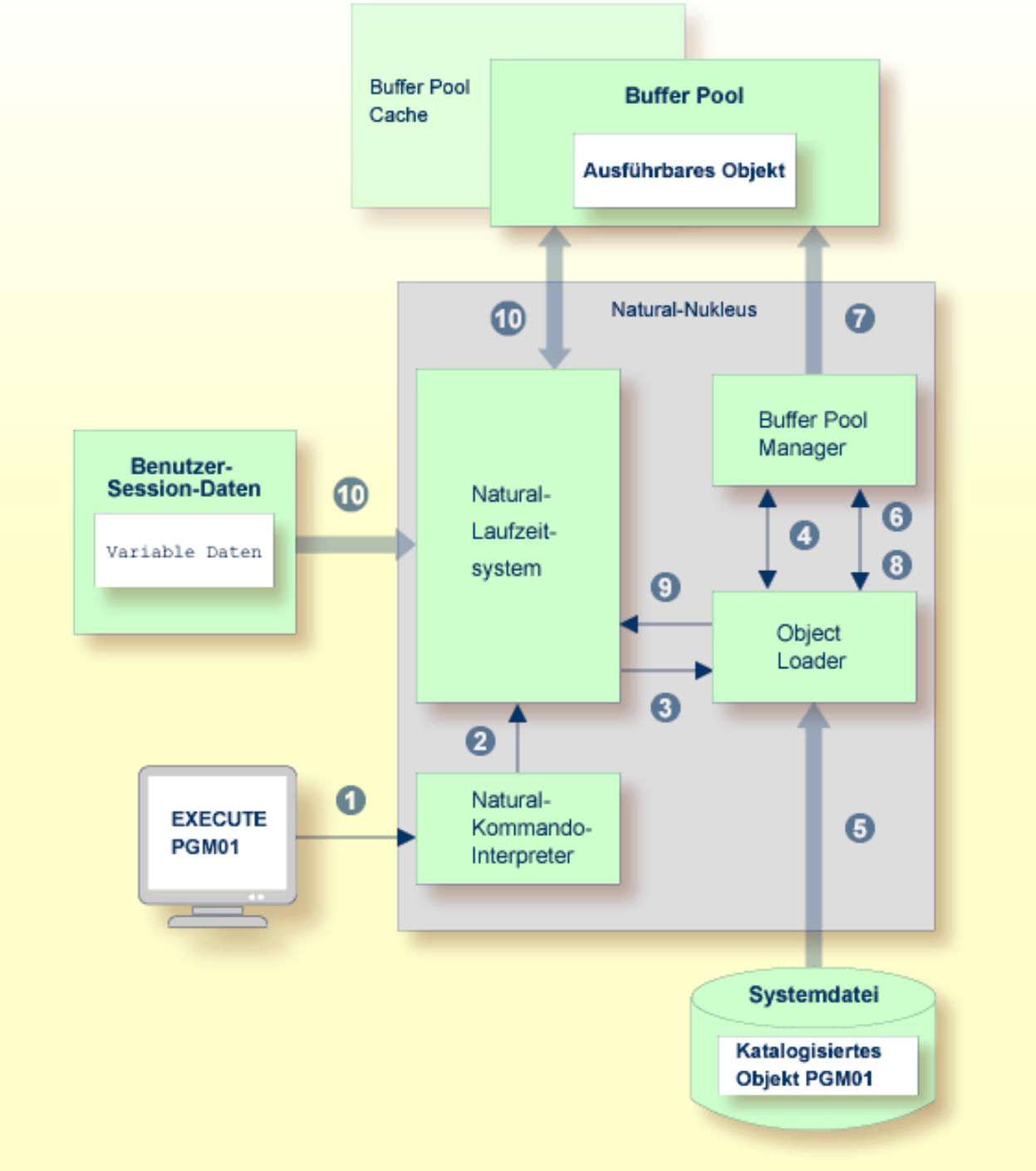
Objekt aus dem Buffer Pool entfernen

Objekte werden aus dem Buffer Pool entfernt, sobald sie nicht mehr von einem Benutzer referenziert werden und der Platz zum Laden neuer Objekte benötigt wird. Dadurch wird die maximale Ausnutzung des im Buffer Pool verfügbaren Speicherplatzes gewährleistet.

Falls ein Buffer Pool Cache vorhanden ist, werden die nicht benutzten Objekte vom Buffer Pool in den Cache-Speicher ausgelagert. Wenn ein nicht benutztes Objekt auch aus dem Cache entfernt wird oder wenn kein solcher Cache existiert, wird das betreffende Objekt erneut von der entsprechenden Natural-Systemdatei geladen, sobald es wieder von einem Benutzer referenziert wird.

Beispiel für das Laden und Ausführen eines Objekts

Das folgende Diagramm veranschaulicht den Vorgang des Ladens und Ausführens eines Natural-Programms:


Buffer Pool Cache
Buffer Pool
Ausführbares Objekt
Natural-Nukleus
10
7
Benutzer-Session-Daten
Variable Daten
10
Natural-Laufzeit-system
Buffer Pool Manager
4
6
8
9
Object Loader
3
2
EXECUTE PGM01
1
Natural-Kommando-Interpreter
5
Systemdatei
Katalogisiertes Objekt PGM01

Legende

1. Ein Benutzer setzt das Natural-Systemkommando `EXECUTE` ab und fordert die Ausführung eines Natural-Objekts des Typs Programm mit dem Namen `PGM01` an.
2. Der Natural-Kommando-Interpreter interpretiert das Kommando und gibt die Anforderung an das Natural-Laufzeitsystem weiter.
3. Das Natural-Laufzeitsystem übergibt die Anfrage an den Object Loader.
4. Der Object Loader weist den Buffer Pool Manager an, den Buffer Pool (und, falls vorhanden, den Buffer Pool Cache) nach dem Objekt `PGM01` zu durchsuchen und die Adresse festzustellen, unter der das Objekt im Buffer Pool bzw. im Cache gespeichert ist.

   Findet der Buffer Pool Manager die Buffer-Pool-Adresse von `PGM01`, wird die Adresse vom Buffer Pool an den Object Loader übergeben und die Ausführung wird mit Schritt 8 fortgesetzt.
   Findet der Buffer Pool Manager die Buffer-Pool-Adresse von PGM01 nicht, meldet der Buffer Pool Manager das negative Ergebnis der Suche an den Object Loader zurück und die Ausführung des Objekts wird mit Schritt 5 fortgesetzt.
5. Der Object Loader ruft das katalogisierte Objekt von `PGM01` aus der entsprechenden Systemdatei ab.
6. Der Object Loader übergibt das katalogisierte Objekt von `PGM01` an den Buffer Pool Manager.
7. Der Buffer Pool Manager lädt `PGM01` in den Buffer Pool.
8. Der Buffer Pool Manager gibt die Buffer-Pool-Adresse von `PGM01` an den Object Loader zurück.
9. Der Object Loader übergibt die Buffer-Pool-Adresse von `PGM01` an das Natural-Laufzeitsystem.
10. Das Natural-Laufzeitsystem interpretiert den kompilierten Code des katalogisierten Objekts `PGM01` und führt ihn aus.

Verwandte Themen

Die folgenden Abschnitte in der *System-Architektur*-Dokumentation nehmen Bezug auf den Natural Buffer Pool:

- ***Natural-Systemdateien***
- ***Natural-Laufzeitsystem*** - *Natural-Nukleus*
- ***Katalogisierte Objekte*** - *Natural-Compiler*, *Natural-Nukleus*

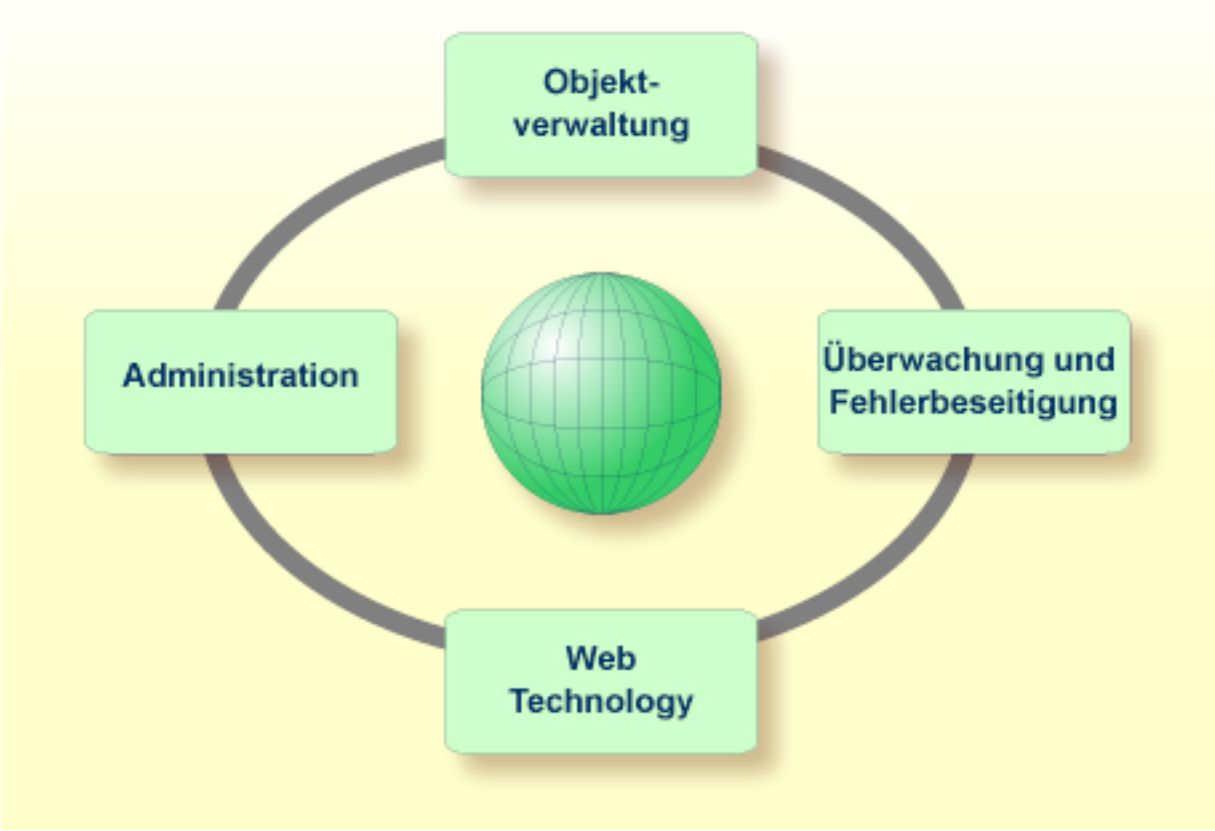
Die folgenden Abschnitte in anderen Natural-Dokumentationen betreffen den Natural Buffer Pool:

- *Natural Buffer Pool - Operations*-Dokumentation
- *Natural Storage Management - Operations*-Dokumentation

6 Natural-Editoren und -Utilities

Natural-Utilities sind Dienstprogramme, die Funktionen zur Verfügung stellen, mit deren Hilfe Sie eine Entwicklungs-, Administrations- und Operations-Umgebung verwalten können.

Das folgende Diagramm zeigt eine Übersicht über die hauptsächlichen Verwendungszwecke der Natural-Editoren und Utilities:



Natural-Editoren bzw. Utilities stehen für folgende Aufgaben zur Verfügung:

- Erstellen und Pflegen von Natural-Objekten (Programme, Subprogramme usw.), **Natural-DDMs** (Datendefinitionsmodule) und Handhabung von Nicht-Natural-Objekten;
- Ausführung von administrativen Funktionen, z.B. Übertragen von Anwendungen von einer Hardware-Plattform auf eine andere, Steuern/Überwachen von Natural Remote Procedure Calls, Erzeugen von Reports und Pflegen von anwendungsspezifischen Fehlermeldungen;
- Überwachen von Anwendungen und Beseitigen von Fehlern (Debugging);
- Anwenden von Web-Technologie, z.B. Zugriffe auf einen Web Server oder Verarbeitung von XML-Dokumenten.

Verwandte Themen:

- ***Natural-DDMs*** *- DBMS-Schnittstelle - Datenbankenzugriff*
- *Editors*-Dokumentation
- *Utilities*-Dokumentation
- *Natural Web Interface - Web Technology*-Dokumentation
- *XML Toolkit - Web Technology*-Dokumentation

7 TP/OS-Schnittstelle

Natural bietet Schnittstellen, über die der Natural-Nukleus-Zugriff auf einen TP-Monitor für die Online-Transaktionsverarbeitung und auf das Betriebssystem (Operating System) für die Batch-Verarbeitung erhält.

Dieses Kapitel behandelt folgende Themen:

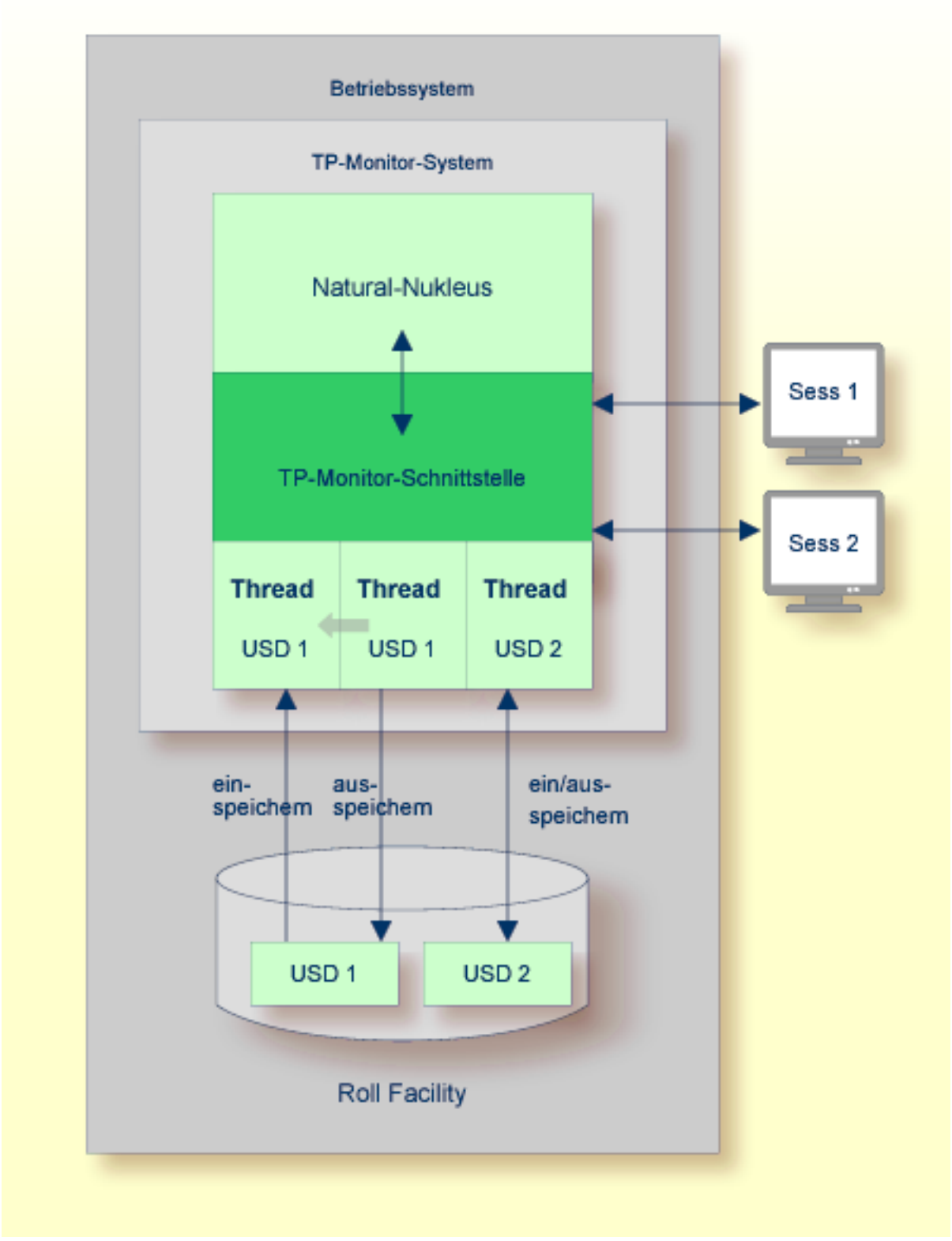
Online-Verarbeitung

In einer TP-Monitor-Umgebung läuft Natural als Standard-TP-Programm. Es arbeitet nach den Regeln, die für Programme gelten, die unter einem TP-Monitor ausgeführt werden.

In einer TP-Monitor-Umgebung bezeichnet man den Arbeitsbereich, in dem **Benutzer-Session-Daten** enthalten sind, als den Natural-Thread. Er wird für die Dauer der Benutzer-Session gepflegt.

Ein Natural-Thread kann, während eine Session auf eine Benutzer-Terminal-Eingabe wartet, in eine Roll Facility (Medium zum Ein-/Ausspeichern) ausgespeichert werden. Durch das Ausspeichern von Threads wird Speicherplatz für andere Benutzer-Sessions freigegeben. Wenn der Benutzer eine Eingabe am Terminal macht, z.B. wenn er EINGABE drückt, wird der Natural-Thread wieder eingespeichert. Ein Thread kann in einen anderen Speicherbereich verlagert werden, wie es im folgenden Diagramm am Beispiel der Benutzer-Session-Daten (`USD`) der Benutzer-Session `Sess 1` gezeigt wird.

Das folgende Diagramm veranschaulicht, wie ein TP-Monitor (hier: Com-plete) die Speicherzuordnung verwaltet, indem er Natural-Threads in eine Roll Facility ein- und ausspeichert:


Betriebssystem
TP-Monitor-System
Natural-Nukleus
TP-Monitor-Schnittstelle
Thread
Thread
Thread
USD 1
USD 1
USD 2
Sess 1
Sess 2
ein-speichern
aus-speichern
ein/aus-speichern
USD 1
USD 2
Roll Facility

Batch-Verarbeitung

Im Batch-Betrieb verarbeitet Natural eine Session, die durch einen Batch Job gestartet wurde. Dabei ist keine Interaktion zwischen dem Rechner und dem Benutzer, der den Batch Job abgeschickt hat, nötig.

Ein Batch Job besteht aus Programmen, die nacheinander ausgeführt werden und dabei sequenziell Eingabedaten erhalten, die von einer Datei gelesen werden. Die Ausgabedaten werden auch in eine Datei geschrieben. Je nach Betriebssystem ist der für einen Batch Job eingerichtete Arbeitsbereich in einer Batch-Region (unter z/OS), einer Partition (unter z/VSE) oder einer Task (unter BS2000/OSD) enthalten. Der Arbeitsbereich enthält Benutzer-Session-Daten, die für die Dauer der Batch-Benutzer-Session verwaltet werden.

Das folgende Diagramm zeigt den Aufbau einer Batch-Verarbeitungsumgebung:


Betriebssystem
Batch-Region/Partition/Task
Natural-Nukleus
Betriebssystem-Schnittstelle
Benutzer-Session-Daten

Natural-TP/OS-Schnittstellen

Informationen zu den bei Natural zur Verfügung stehenden TP-Monitor-Schnittstellen und zur Benutzung von Natural mit einem TP-Monitor finden Sie in den entsprechenden Abschnitten in der *TP Monitor Interfaces*-Dokumentation:

- *Using Natural with TP Monitors*
- *Natural under CICS*
- *Natural under Com-plete/SMARTS*
- *Natural under IMS TM*
- *Natural under TIAM*
- *Natural under TSO*
- *Natural under openUTM*
- *Natural under VM/CMS*

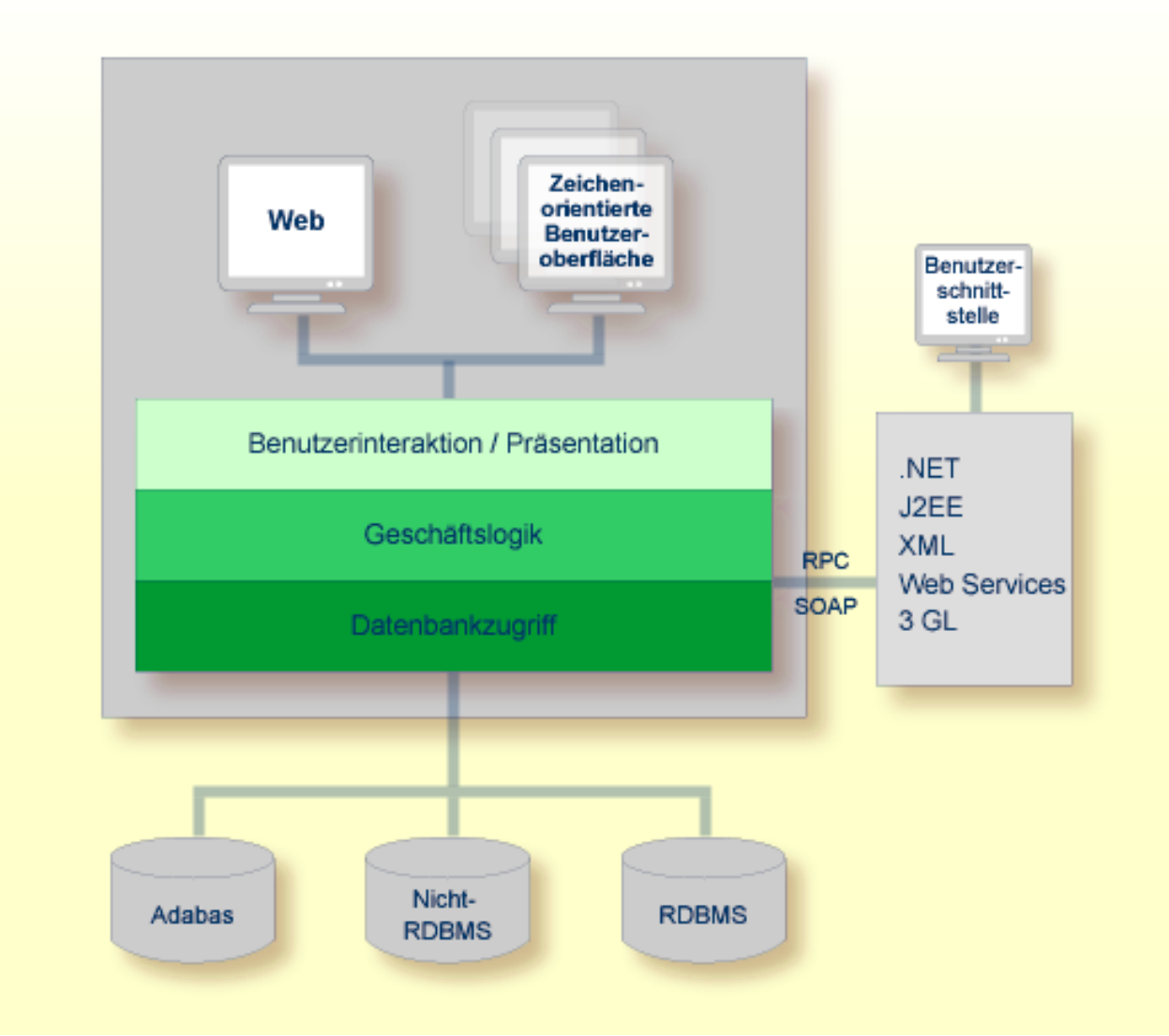
Informationen zu Natural im Batch-Betrieb finden Sie im Abschnitt *Natural in Batch Mode* in der *Operations*-Dokumentation.

8 Benutzerschnittstelle

Natural unterstützt verteilte Anwendungen mit mehrschichtiger Architektur: Darstellungsschicht, fachliche Anwendungslogikschicht und Datenbankzugangsschicht. Eine Natural-Anwendung ist modular aufgebaut und bietet die Objektarten, die für jede Anwendungsschicht erforderlich sind.

Natural ermöglicht die Programmierung der verschiedenartigsten Benutzerschnittstellen: webbasierte Schnittstellen (z.B. HTML oder XML), prozessgesteuerte Anwendungen mit Zeichenoberflächen (CUIs) und ereignisgesteuerte Anwendungen mit grafischen Benutzeroberflächen (GUIs).

Außerdem kann Natural mit Benutzeroberflächen interagieren, die mit Nicht-Natural-Umgebungen, z.B. J2EE and .NET, verbunden sind. Verbindungen zwischen Natural- und Nicht-Natural-Umgebungen werden beispielsweise mittels Remote Procedure Call (RPC) oder einer SOAP-Anforderung hergestellt, über die es für Programme auf einem Client-Computer möglich ist, ein Natural-Objekt des Typs Subprogramm auf einem Natural-RPC-Server auszuführen.


Web
Zeichen-
orientierte
Benutzer-
oberfläche
Benutzer-
schnitt-
stelle
Benutzerinteraktion / Präsentation
Geschäftslogik
Datenbankzugriff
RPC
SOAP
.NET
J2EE
XML
Web Services
3 GL
Adabas
Nicht-
RDBMS
RDBMS

9 Druckdateien und Arbeitsdateien

Druckdateien (Print Files) und Arbeitsdateien (Work Files) sind „Datenbehälter“ für die dauerhafte oder zwischenzeitliche Speicherung von Daten, die in Natural-Objekten verarbeitet werden. Bei diesen „Datenbehältern“ handelt es sich beispielsweise um physische Dateien, Members von partitionierten Datasets oder Tape Datasets, die entweder vom Betriebssystem oder dem TP-Monitor zur Verfügung gestellt werden. Der Zugriff von Natural auf die Druckdateien und Arbeitsdateien erfolgt sequenziell.

Eine Druckdatei dient dazu, Report-Daten und Druckersteuerzeichen zu speichern, die für die Ausgabe eines Reports auf einem Drucker benötigt werden.

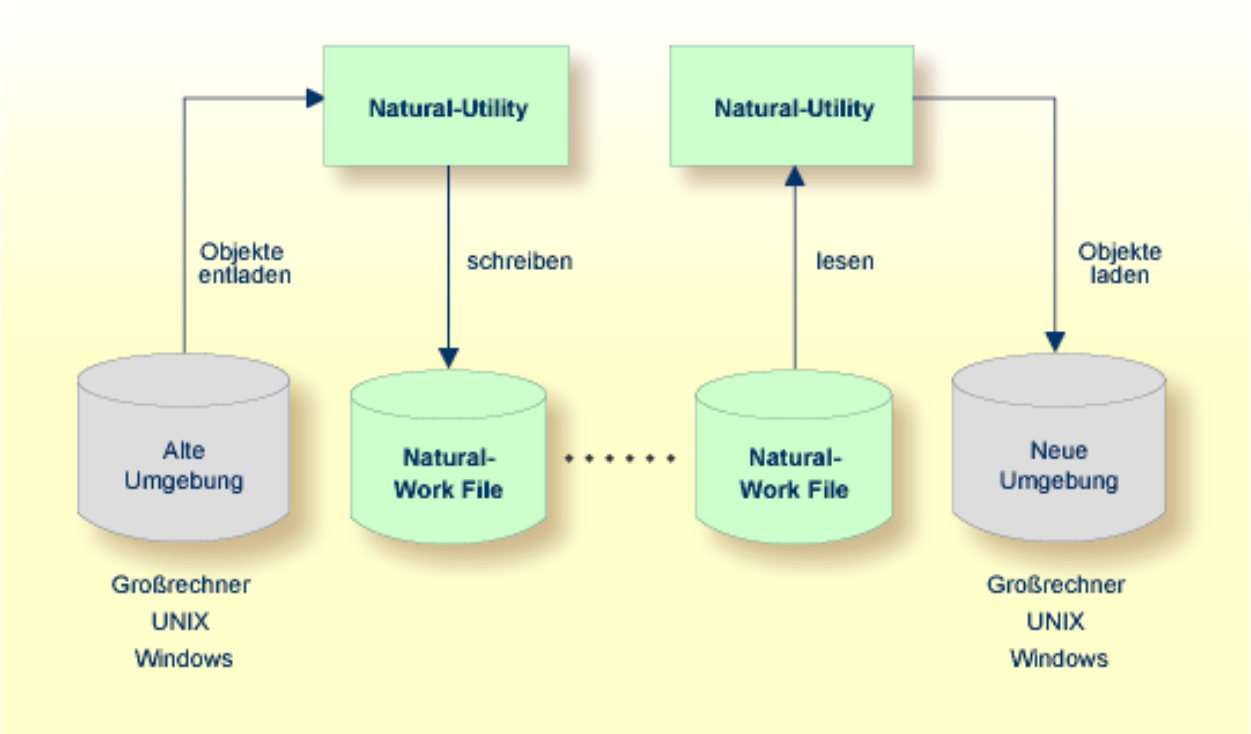
Eine Arbeitsdatei wird verwendet, um Daten zu speichern, die zwischen Natural-Objekten untereinander oder zwischen einem Natural-Objekt und einem in einer anderen Programmiersprache, z.B. COBOL, geschriebenen Objekt ausgetauscht werden.

Druckdateien und Arbeitsdateien können unter Berücksichtigung der Konventionen des installierten Betriebssystems bzw. TP-Monitors zur Batch- oder Online-Verarbeitung verwendet werden.

Der folgende Abschnitt behandelt folgende Themen:

Objekte mit Hilfe von Arbeitsdateien übertragen

Das folgende Diagramm veranschaulicht, wie die in einer Arbeitsdatei (Work File) enthaltenen Daten verwendet werden können, um Natural-Objekte zwischen verschiedenen Natural-Umgebungen auf unterschiedlichen Plattformen zu übertragen. Dies geschieht mit Hilfe eines Dienstprogramms, einer Natural-Utility (z.B. der Object Handler), mit dem die in der Ursprungsumgebung vorhandenen Objekte in ein Work File entladen und von dort in die Zielumgebung geladen werden. Falls erforderlich, wird ein Anwendungsprotokoll, z.B. FTP, benutzt, um Work Files von Ursprungs- in Zielumgebungen zu übertragen.



Druckdateien und Arbeitsdateien, Definition und Zugriff

Druckdateien und Arbeitsdateien werden logisch für eine Natural-Umgebung definiert und können durch entsprechende Angaben in einem Natural-Parameter und/oder in Control-Statements des zugrundeliegenden Betriebs- oder TP-Monitorsystems einer Datei bzw. einem Drucker physisch zugeordnet werden. Die Zuordnungen können für die aktuelle Session zur Laufzeit geändert werden.

Die Daten werden unter Verwendung der entsprechenden Natural-Statements in eine Arbeitsdatei geschrieben und von dort gelesen bzw. in eine Druckdatei geschrieben.

10 Natural-Systemdateien

In einer Natural-Systemdatei werden Natural-Objekte gespeichert, die entweder zu Benutzeranwendungen oder zu von der Software AG gelieferten Natural-Systemanwendungen (z.B. Utilities) gehören.

Zu den in einer Systemdatei gespeicherten Natural-Objekten gehören **katalogisierte Objekte** und **Source-Objekte**, siehe Beschreibung des Natural-Compilers, Abschnitt **Natural-Nukleus**. Natural-Systemdateien werden in Datenbankdateien oder Speichersystemen wie z.B. Adabas und VSAM gespeichert. Je nach Systemumgebung können die Natural-Systemdateien in unterschiedlichen Datenbankdateien oder Speichersystemen untergebracht sein.

Der Zugriff auf eine Natural-Systemdatei erfolgt, wenn ein Benutzer ein Natural-Objekt liest oder ändert. Außerdem wird auf eine Natural-Systemdatei zugegriffen, wenn ein Objekt zur anschließenden Ausführung in den Buffer Pool geladen wird (siehe Abschnitt ***Natural Buffer Pool***).

Dieser Abschnitt behandelt folgende Themen:

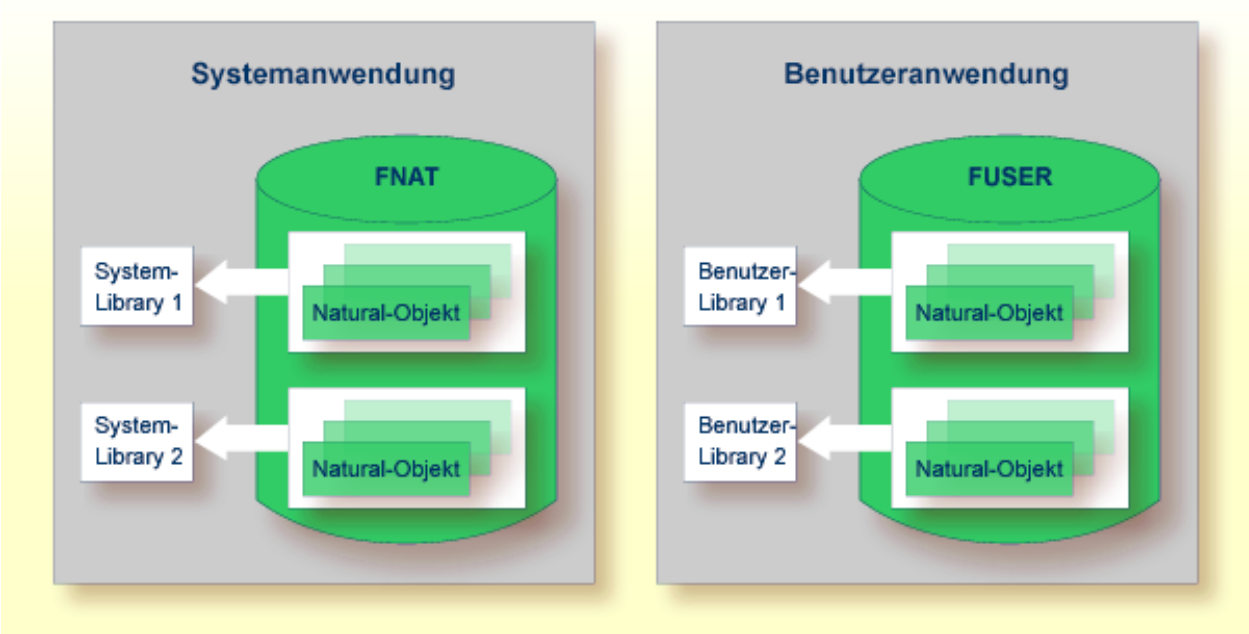
Systemdatei-Arten

In der folgenden Tabelle sind diejenigen Natural-Systemdateien aufgeführt, die normalerweise in einer Natural-Umgebung vorhanden sind. Die Verfügbarkeit der Systemdateien und deren Inhalte hängen von den Software AG-Produkten ab, die zusätzlich zu Basis-Natural installiert sind.

Systemdatei	Verfügbar mit	Datei-Inhalt
FNAT	Basis-Natural	Alle für Natural-Systemanwendungen benötigten Objekte.
FUSER	Basis-Natural	Für Benutzeranwendungen benötigte benutzerspezifische Objekte.
FDIC	Basis-Natural	Natural-DDMs (Datendefinitionsmodule). Wenn Predict installiert ist, enthält FDIC außerdem noch Daten für das Predict-Datendiktionärsystem. Wenn der Natural Development Server installiert ist, enthält FDIC außerdem noch Anwendungsdaten und Informationen über die Sperrung von Objekten.
FSEC	Natural Security	Für Sicherheitseinstellungen benötigte Steuerungs-/Überwachungsinformationen.
FSPOOL	Natural Advanced Facilities	Steuerungs- und Spool-Informationen, die zum Ausgeben eines Reports auf dem Bildschirm oder einem Drucker und zum Erstellen von statistischen Druckinformationen benötigt werden.
Scratch-Pad	Basis-Natural	Daten die nicht explizit als Natural-Objekte in einer anderen Systemdatei gespeichert sind.

Libraries in Systemdateien

Die in den Systemdateien FNAT und FUSER enthaltenen Natural-Objekte sind in logische Konstrukte unterteilt, die als Natural Libraries (Bibliotheken) bezeichnet werden; siehe folgendes Diagramm:



Verwandte Themen:

- ***Katalogisiertes Objekt*** und ***Source-Objekt*** - *Natural-Compiler, Natural-Nukleus*
- *Natural Security*-Dokumentation
- *Natural Advanced Facilities*-Dokumentation
- ***Natural-DDMs*** - *DBMS Interface - Database Access*
- *Predict*-Dokumentation
- *Natural Development Server*-Dokumentation

11 DBMS-Schnittstelle - Datenbankzugriff

Natural bietet Schnittstellen zu verschiedenen Datenbankverwaltungssystemen (DBMS), über die der Zugang zu Daten möglich ist, die z.B. in einer Adabas-Datenbank, einem relationalen Datenbankverwaltungssystem (RDBMS) und/oder in einem Nicht-RDBMS gespeichert sind.

Natural-Anwendungen können gleichzeitig auf Benutzerdaten in mehreren DBMS zugreifen. Im Gegensatz zum Zugriff auf **Druckdateien und Arbeitsdateien** (siehe entsprechenden Abschnitt), der sequenziell erfolgt, kann der Zugriff auf die in einem DBMS gespeicherten Daten wahllos erfolgen.

Zu den von Natural unterstützten DBMS gehören DB2, DL/I und VSAM.

Dieser Abschnitt enthält Informationen zu den von Natural unterstützten DBMS und beschreibt, wie prinzipiell aus einer Natural-Anwendung heraus auf Daten in einer Datenbank zugegriffen wird:

Von Natural unterstützte Datenbankverwaltungssysteme

Natural bietet Schnittstellen zu den folgenden Datenbankverwaltungssystemen (DBMS):

- **Adabas**
- **DL/I**
- **DB2**
- **VSAM**

Adabas

Die Natural Adabas-Schnittstelle ermöglicht den Zugriff auf Daten, die in einer Adabas-Datenbank gespeichert sind.

Adabas ist das DBMS der Software AG. Adabas unterstützt sowohl relationale als auch geschachtelte Datenstrukturen.

Die Natural Adabas-Schnittstelle ist ein integrierter Bestandteil von Natural. Sie gestattet den Zugriff auf Adabas-Datenbanken auf lokalen Rechnern. Für den Remote-Zugang wird zusätzliche Routing- und Kommunikationssoftware benötigt, z.B. Entire Net-Work von der Software AG. Der Typ der Host-Maschine, auf der die Adabas-Datenbank läuft, ist in jedem Fall für Natural immer transparent.

Verwandte Themen:

- *Daten in einer Adabas-Datenbank aufrufen - Leitfaden zur Programmierung*
- *Adabas*-Dokumentation

DL/I

Die Natural DL/I-Schnittstelle ermöglicht den Zugriff auf Daten, die in einer IMS DB-Datenbank gespeichert sind.

DL/I ist die Datenverarbeitungssprache für IMS DB von IBM. IMS ist ein Datenbank- und Transaktionsverwaltungssystem von IBM.

Verwandtes Thema:

- *Natural for DL/I*-Dokumentation

DB2

Die Natural DB2-Schnittstelle ermöglicht den Zugriff auf Daten, die in einer DB2 UDB von IBM für z/OS-Betriebssysteme gespeichert sind.

Verwandtes Thema:

- *Natural for DB2*-Dokumentation

VSAM

Die Natural VSAM-Schnittstelle ermöglicht den Zugriff auf Daten, die in VSAM-Datasets gespeichert sind.

VSAM ist ein IBM-Zugriffsverfahren, mit dem Datensätze verschiedener Dataset-Organisationsformen verwaltet werden können: Key-Sequenced Datasets, Entry-Sequenced Datasets oder Relative-Record Datasets.

Verwandtes Thema:

- *Natural for VSAM*-Dokumentation

Natural-Datenbankabfragesprache

Die Natural-Datenbankabfragesprache (Data Manipulation Language - DML) bietet eine gemeinsame Datenzugriffssyntax für alle von Natural unterstützten Datenbankverwaltungssysteme (DBMS). Durch die Natural-DML können Natural-Objekte unter Verwendung derselben Sprach-Statements (DML-Statements), z.B. `FIND`, `READ`, `STORE` und `DELETE`, auf verschiedenartige DBMS zugreifen.

Natural bestimmt die DBMS anhand seiner Konfigurationsdateien und übersetzt die DML-Statements in datenbankspezifische Befehle; d.h., Natural erzeugt Direktkommandos für Adabas-

Datenbanken, SQL-Statement-Strings und Host-Variablen-Strukturen für RDBMS, die SQL verwenden.

Ein Natural-Objekt, in dem ein DML-Statement verwendet wird, das *keine* datenbankspezifische Klausel enthält, kann gegen unterschiedliche DBMS ausgeführt werden. Ein DML-Statement, das eine datenbankspezifische Klausel enthält, muss geändert warden, bevor es auf einem andersartigen DBMS verwendet werden kann. In der *Statements*-Dokumentation sind datenbankspezifische Hinweise enthalten.

Verwandtes Thema:

- *Statements*-Dokumentation

Spezielle SQL-Statements

Zusätzlich zu den DML-Statements bietet Natural einen Satz spezieller SQL-Statements zur ausschließlichen Verwendung in Verbindung mit RDBMS. Dies sind die Statements `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `COMMIT` und `ROLLBACK`, die in der *Statements*-Dokumentation beschrieben sind.

Der Satz spezieller SQL-Statements wird vervollständigt durch Flexible SQL und Funktionen zum Arbeiten mit Stored Procedures. Flexible SQL gestattet die Benutzung frei wählbarer Syntax.

Verwandte Themen:

- *SQL Statements - Statements*-Dokumentation
- *Flexible SQL - Statements*-Dokumentation

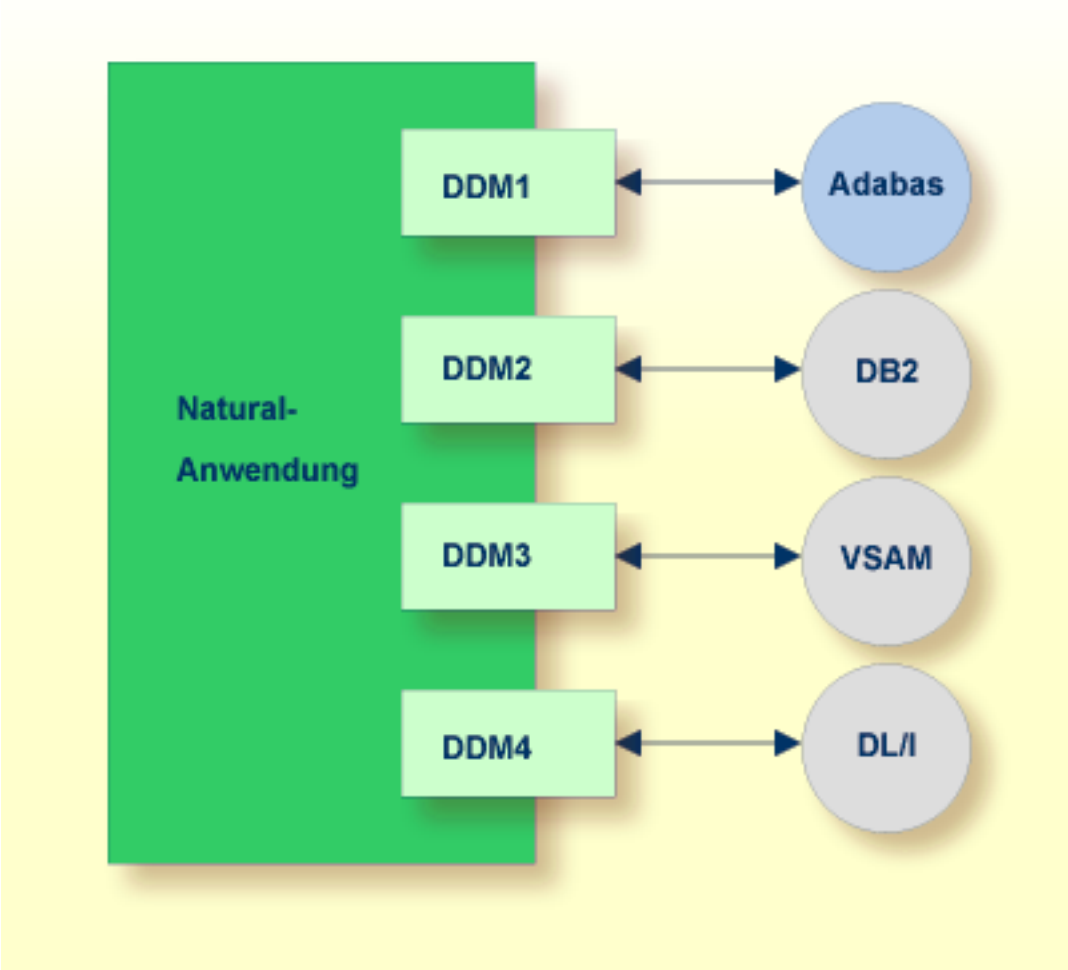
Natural-DDMs

Natural bietet ein Objekt, das als DDM (Datendefinitionsmodul) bezeichnet wird und den komfortablen und transparenten Zugriff auf Datenbankdateien auf unterschiedlichen DBMS ermöglicht.

Ein DDM ist eine logische Sicht auf eine physische Datenbankdatei. Das DDM enthält Informationen zu den einzelnen Feldern der Datei, die für die Verwendung dieser Felder in einem Natural-Objekt relevant sind.

Ein DDM stellt die Verbindung zwischen den Natural-Datenstrukturen und den Datenstrukturen in dem zu verwendenden DBMS her. Bei einer solchen Struktur kann es sich um eine Tabelle in einem RDBMS, eine Datei in einer Adabas-Datenbank oder einen VSAM Dataset handeln. Das DDM verbirgt somit die wirkliche Struktur der Datenbank, auf die zugegriffen wird, gegenüber der Natural-Anwendung.

Das folgende Diagramm veranschaulicht, wie Natural aus einer einzelnen Anwendung heraus auf mehrere, unterschiedliche Datenbanken zugreifen kann, indem es Referenzen auf DDMs verwendet, die die spezifischen Datenstrukturen im jeweiligen DBMS darstellen:

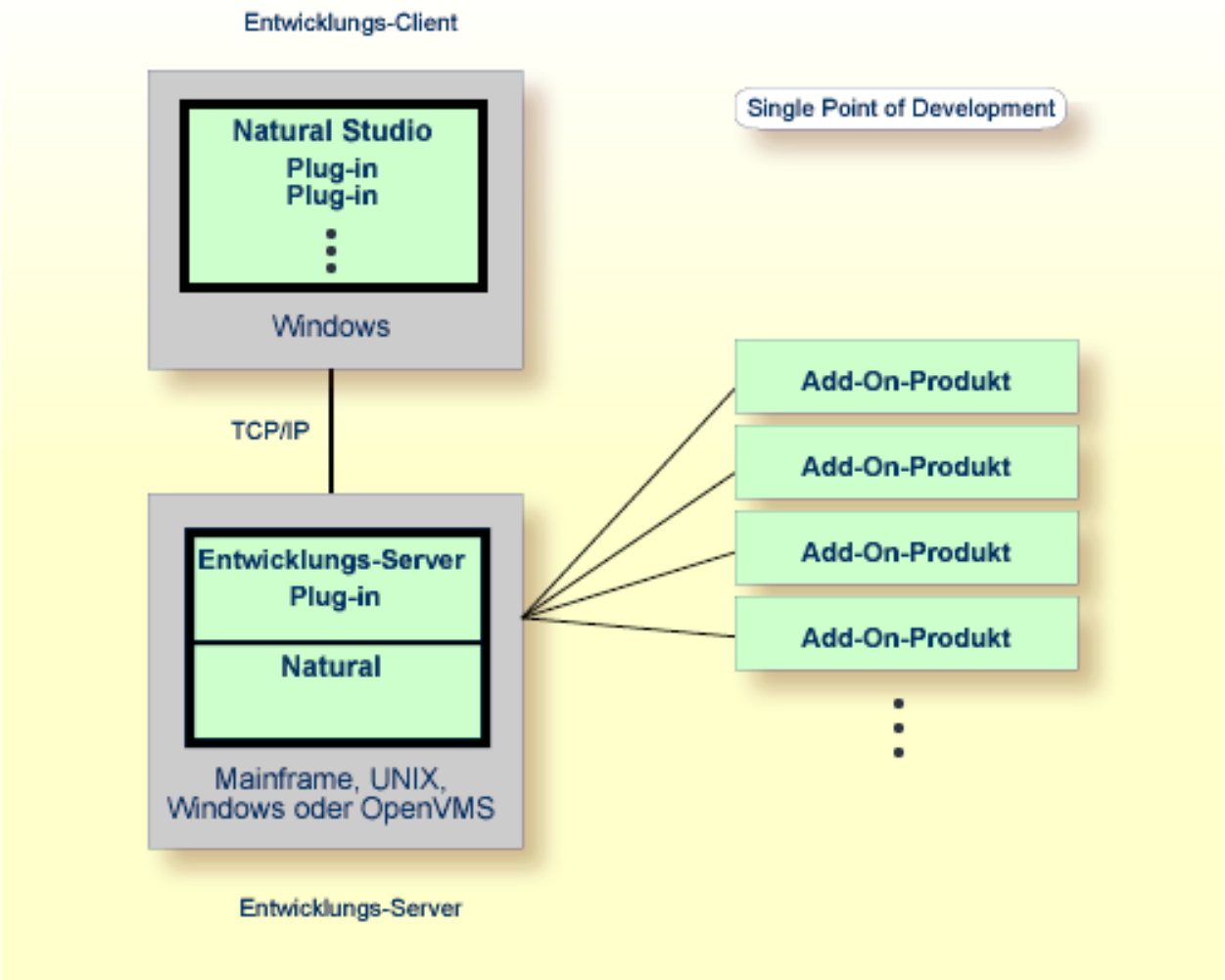


12 Natural-SPoD-Architektur

Dieser Abschnitt beschreibt die Systemarchitektur des Natural Single Point of Development (SPoD).

SPoD ermöglicht eine zentralisierte Anwendungsentwicklung aus einer einzelnen Windows-Umgebung heraus. Sie können die Funktionalität von Natural Studio (Bestandteil von Natural für Windows) benutzen, um Natural-Anwendungen in einer Remote-Umgebung zu entwickeln, die sich auf einer Großrechner- oder UNIX-Plattform befindet.

SPoD basiert auf einem Client/Server-Konzept, das eine einzige Entwicklungsumgebung für alle Plattformen ermöglicht. Das folgende Diagramm veranschaulicht dieses Konzept und die Hauptbestandteile der SPoD-Architektur:



Development Client

Natural für Windows dient als Remote-Development-Desktop-Client für die Zielumgebung auf einer Großrechner oder einer UNIX-Plattform. Zum Desktop-Client gehört auch Natural Studio, die zentrale Workstation mit grafischer Benutzeroberfläche, auf der die Benutzer Anwendungen entwickeln können. Aus Natural Studio können sie alle Natural-Funktionen ausführen, die bei der Remote-Entwicklung benötigt werden.

Development Server

Das Natural-Development-Server-Plug-In ermöglicht die Remote-Entwicklung mit einem Natural, das in einer Zielumgebung auf einer Großrechner- oder UNIX-Plattform installiert ist. Das Natural auf der Zielplattform und das Natural-Development-Server-Plug-In bilden zusammen den Development Server.

Add-On-Produkt

Natural Studio bietet Plug-Ins, die installiert werden können, um ein oder mehrere Natural-Add-On-Produkte (z.B. Predict) in eine SPoD-Umgebung zu integrieren. Um ein Natural Studio Plug-In benutzen zu können, muss das entsprechende Add-On-Produkt in der Development-Server-Umgebung installiert sein.

Security

Um die Natural-Development-Server-Umgebung und die Natural-Base-Applications und Compound-Applications zu schützen, können Sie Natural Security einsetzen. Weitere Informationen siehe *Natural Security*-Dokumentation.

Verwandtes Thema:

- *Natural Single Point of Development*-Dokumentation