

# Caché Technology-Guide



Innovationen von **InterSystems**

# Inhaltsverzeichnis

EINLEITUNG	3
------------	---

## KAPITEL 1

<b>DATENMODELLIERUNG – RELATIONALER ODER OBJEKTZUGRIFF</b>	<b>6</b>
RELATIONALE TECHNOLOGIE	7
OBJEKTTECHNOLOGIE UND OBJEKTDATENBANKEN	8
OBJEKT- ODER RELATIONALER ZUGRIFF	8
DAS CACHÉ OBJEKTDATENMODELL UND OBJEKTPROGRAMMIERUNG: EIN ÜBERBLICK	8
SCHLÜSSELBEGRIFFE DER OBJEKTORIENTIERUNG	10
WAS SPRICHT FÜR OBJEKTE ZUR DATENMODELLIERUNG?	11
SPEICHERUNG VON OBJEKTEN...PLUS RELATIONALER ZUGRIFF	11

## KAPITEL 2

<b>DER MULTIDIMENSIONALE CACHÉ-DATENSERVER</b>	<b>12</b>
INTEGRIERTER DATENBANKZUGRIFF	12
DAS MULTIDIMENSIONALE DATENMODELL	13
SQL-ZUGRIFF	16
OBJEKTE IN CACHÉ	18
TEXTSUCHE MIT WORTERKENNUNG	20
TRANSAKTIONALE BITMAP-INDIZIERUNG	22
DAS ENTERPRISE CACHE PROTOCOL FÜR VERTEILTE SYSTEME	24
FEHLERTOLERANZ	26
SICHERHEITSMODELL	29

## KAPITEL 3

<b>DER CACHÉ-APPLIKATIONSSERVER</b>	<b>32</b>
DIE CACHÉ VIRTUAL MACHINE UND SKRIPTSPRACHEN	32
CACHÉ OBJECTSCRIPT	34
BASIC	40
MV BASIC	42
C++	42
JAVA	42
CACHÉ UND JALAPEÑO	44
CACHÉ UND .NET	46
CACHÉ UND XML	47
CACHÉ UND WEB SERVICES	48
CACHÉ UND MULTIVALUE	49

## KAPITEL 4

<b>SCHNELLE WEB-ANWENDUNGEN SCHNELL ENTWICKELT MIT CACHÉ SERVER PAGES</b>	<b>52</b>
DAS SERVER PAGE-MODELL VON CACHÉ	54
DIE KLASSENARCHITEKTUR VON WEBSEITEN	56
MEHRERE ENTWICKLUNGSSTRATEGIEN	56
CSP-DATEIEN	57
HYPER-EVENTS	58
ZEN UND KOMPONENTENBASIERTE WEBSEITEN	59

### DIE POSTRELATIONALE ÄRA DER DATENVERARBEITUNG HAT BEGONNEN

Das Aufkommen der relationalen Datenbanken stellte vor 30 Jahren eine große Innovation dar. Statt in monolithischen Altdatenbanken mit proprietärem Datenschema konnten Daten nun im einheitlichen Tabellenformat gespeichert werden und standen für jeden zum Abruf bereit, der SQL beherrschte. Entsprechend erfolgreich waren die relationalen Datenbanken und machten SQL schnell zum allgemeinen Standard für den Zugriff auf Datenbanken. Aus heutiger Sicht betrachtet, weisen relationale Datenbanken allerdings einige Defizite auf, die ihrem Einsatzbereich Grenzen setzen – vor allem hinsichtlich Performance und Skalierbarkeit, der Bedienerfreundlichkeit sowie der Eignung für moderne Entwicklungstechnologien.



Die Nutzung und Komplexität von Computeranwendungen nimmt exponentiell zu. Die heutigen Systeme stellen in zunehmendem Maße Verarbeitungsanforderungen, die die Möglichkeiten der relationalen Technologie übersteigen. Viele Schlüsselanwendungen, die eine hohe Performance und Skalierbarkeit erfordern, wurden überhaupt nicht auf relationale Datenbanken portiert, und selbst einfache Anwendungen stoßen heute immer häufiger an die Grenzen der traditionellen relationalen Technologie.

Der “Impedance Mismatch” zwischen relationalen Datenbanken und den heutigen Entwicklungstechnologien ist zu einem gravierenden Problem geworden – er macht die Entwicklung komplexer und erhöht das Risiko zu scheitern. Während die Einfachheit der tabellarischen Strukturen eine elegante Abfragesprache (SQL) fördert, erweist es sich als schwierig, die Datenstrukturen der realen Welt dermaßen vereinfacht in Zeilen und Spalten zu zerlegen. Das Ergebnis ist eine riesige Anzahl an Tabellen, deren Beziehungen schwer nachzuvollziehen und mühsam zu nutzen sind – Zeilen und Spalten sind einfach, aber ständig „Left Outer Joins“, „Stored Procedures“ und „Trigger“ programmieren zu müssen, macht die Entwicklung kompliziert.

Moderne Anwendungen werden normalerweise objektorientiert erstellt, weil sich so Informationen schneller und intuitiver beschreiben und nutzen lassen. Das beschleunigt die Entwicklung und erhöht die Zuverlässigkeit. Leider sind Objekte von der Grundkonzeption her nicht mit relationalen Datenbanken kompatibel. Die Vorteile der objektorientierten Technologie verblassen schnell, wenn die entstehenden Datenbankobjekte in ein zweidimensionales relationales Modell gezwängt werden müssen.

*Die Anforderungen moderner transaktionsverarbeitender Anwendungen überfordern die Möglichkeiten relationaler Technologie – sie sollen umfangreiche Netzwerke umspannen, Tausende von Benutzern in exzellenter Performance bedienen, Web-kompatibel und kostengünstig zu betreiben sein. Und sie müssen schnell entwickelt werden!*

## Was ist Caché?

---

InterSystems Caché gehört zu einer neuen Generation extrem hochleistungsfähiger Datenbanktechnologie. Caché vereint eine objektorientierte Datenbank, Hochleistungs-SQL und einen multidimensionalen Datenzugriff – wobei mit allen Methoden gleichzeitig auf dieselben Daten zugegriffen werden kann. Die Daten werden in einem einzigen integrierten Data Dictionary nur ein Mal beschrieben und stehen sofort für alle Zugriffsmethoden zur Verfügung. Im Vergleich zu relationaler Technologie zeichnet sich Caché durch eine exzellente Performance, höchste Skalierbarkeit, kurze Entwicklungszeiten und einfache Bedienbarkeit aus.

Aber Caché ist viel mehr als nur eine Datenbanktechnologie. So verfügt Caché über einen Applikationsserver für modernste objektorientierte Programmierung, kann eine breite Palette weiterer Technologien integrieren und enthält eine Hochleistungslaufzeitumgebung mit einzigartigen Caching-Mechanismen.

Caché unterstützt verschiedene eingebaute Skriptsprachen: Caché ObjectScript, eine leistungsstarke und einfach zu erlernende objektorientierte Programmiersprache, Caché Basic, eine Obermenge der weit verbreiteten Programmiersprache Basic mit Erweiterungen für einen leistungsstarken Datenzugriff und Objekttechnologie sowie Caché MVBasic, eine Basic-Variante, die von MultiValue-Anwendungen (manchmal auch als Pick-Anwendungen bezeichnet) verwendet wird. Andere Sprachen wie Java, C# und C++ werden über direkte Call-in-Schnittstellen bzw. über ODBC, JDBC, .NET u.a. unterstützt. Zudem ermöglicht eine Objekt-Schnittstelle den Zugriff auf die Caché-Datenbank und andere Elemente von Caché als Eigenschaften und Methoden.

Auch in einem anderen Bereich geht Caché über die Bandbreite herkömmlicher Datenbanken hinaus: Es bringt nämlich eine umfangreiche Umgebung zur Entwicklung ausgefeilter Browser-basierter (Web-)Anwendungen mit. Mit der Caché-Server-Pages-(CSP)-Technologie lassen sich dynamisch erzeugte Webseiten schnell erstellen und ausführen. Dadurch können Tausende von Web-Benutzern gleichzeitig auf die Datenbankanwendungen zugreifen, selbst wenn diese auf preiswerter Hardware laufen.

Für Anwendungen, die nicht auf Browser-Technologie basieren, kann die Benutzeroberfläche mit gängigen Front-End-Werkzeugen wie Java, .NET, Delphi oder C++ entwickelt werden. Die besten Ergebnisse (schnellste Programmierung, höchste Performance und geringster Wartungsaufwand) erhält man in der Regel dann, wenn die gesamte restliche Entwicklungsarbeit in Caché vorgenommen wird. Aber Caché bietet auch einen äußerst hohen Grad an Interoperabilität mit anderen Technologien und unterstützt alle gängigen Entwicklungswerkzeuge, so dass ein breites Spektrum an Entwicklungsmethoden zur Verfügung steht.



## Kapitel 1: Datenmodellierung – relationaler oder Objektzugriff

Sobald ein Entwickler mit dem Design einer neuen Anwendung beginnt, steht auch die Grundsatzentscheidung über die Datenmodellierung an – in der Regel eine Entscheidung zwischen dem traditionellen Modell relationaler Tabellen und dem moderneren Ansatz der Modellierung als Objekte. Vor dem Hintergrund komplexer Datenoperationen halten viele Entwickler die Modellierung mit Objekten für den effizienteren Ansatz.

Bei der Übertragung einer bestehenden Anwendung nach Caché geht es im ersten Schritt natürlich darum, das vorhandene Datenmodell zu migrieren. Hierfür stehen einfache Verfahren zur Verfügung, mit denen verschiedene relationale oder objektorientierte Darstellungen und Daten importiert werden können. Als Ergebnis erhält man eine Datendefinition, die den Standards von Caché entspricht, und Daten, auf die gleichzeitig als Objekte, relationale Tabellen und multidimensionale Arrays zugegriffen werden kann.

Caché unterstützt grundsätzlich sowohl SQL- als auch Objektzugriff und es gibt gute Anwendungsfälle für beide. Um zu verstehen, wann sich welcher Zugriff anbietet und warum sich Entwickler heute vorzugsweise für die Objektmodellierung entscheiden, ist es sinnvoll, sich kurz mit der Entscheidungsgeschichte beider Methoden auseinander zu setzen.



## RELATIONALE TECHNOLOGIE

In der Frühzeit der EDV war die Datenverarbeitung Aufgabe riesiger Mainframes und der Zugriff auf Daten war üblicherweise professionellen Informatikern vorbehalten. Die Datenbanken waren häufig individuelle Eigenentwicklungen und die Auswertung und Aktualisierung der Daten verlangte profunde Kenntnisse der gesamten Datenbankorganisation. Anwender mit Bedarf an speziellen Auswertungen waren auf die Unterstützung einer überlasteten DV-Abteilung angewiesen und erhielten ihre Listen deshalb selten rechtzeitig, um wichtige Entscheidungen darauf zu stützen.

Relationale Technologie entstand ursprünglich in den 1970er-Jahren auf dem Mainframe, blieb aber weitgehend im Forschungsstadium, bis sich in den 1980ern die Minicomputer durchsetzten. Mit dem Siegeszug des PC begann dann das benutzerzentrierte Zeitalter der EDV mit benutzerfreundlicheren Auswertungswerkzeugen auf der Basis von SQL – der Abfragesprache, die mit der relationalen Technologie Einzug gehalten hatte. Von nun an konnten Benutzer ihre eigenen Listen und ad-hoc-Abfragen aus der Datenbank erzeugen und relationale Datenbanken erlebten einen regelrechten Boom.

SQL erlaubt die Verwendung einer konsistenten Sprache, um Abfragen auf ein großes Spektrum an Daten auszuführen. Dabei betrachtet SQL alle Daten in einem sehr einfachen, standardisierten

Format: als zweidimensionale Tabelle mit Zeilen und Spalten. Dieses simple Datenmodell befriedigte die Bedürfnisse der Anwender nach einfachen Auswertungen sehr elegant, hatte aber auch einen stolzen Preis. Denn es liegt in der Natur der Sache, dass sich komplexe Datenstrukturen der realen Welt nicht ganz so einfach in Zeilen und Spalten abbilden lassen. So sind Daten häufig auf viele Tabellen verteilt, die erst wieder mühsam verbunden werden müssen, um selbst einfachste Aufgaben zu lösen. Genau hierin liegen die beiden Hauptprobleme: a) Das Schreiben von Abfragen (Queries) wird zu einer schwierigen und fehlerträchtigen Aufgabe, weil viele Tabellen miteinander verbunden werden müssen (oftmals durch komplexe „Outer Joins“). b) Relationale Datenbanken mit komplexen Daten stellen enorme Ansprüche an die Verarbeitungskapazität.

SQL ist heute der Standard für Datenbank-Interoperabilität und Auswertungswerkzeuge. Gerade weil die Wurzeln von SQL in relationaler Datenbanktechnologie liegen, ist es besonders wichtig zu verstehen, dass es nicht zwangsläufig nur mit relationalen Datenbanken funktioniert. So unterstützt Caché die Abfragesprache SQL, obwohl es auf einer wesentlich leistungsfähigeren multidimensionalen Datenbanktechnologie gründet, und implementiert SQL-Erweiterungen zur Unterstützung von Objekten.



## OBJEKTTECHNOLOGIE UND OBJEKTDATENBANKEN

Objektprogrammierung und Objektdatenbanken sind das praktische Ergebnis wissenschaftlicher Arbeit über die Abläufe menschlichen Denkens. Untersuchungen haben gezeigt, dass unser Gehirn in der Lage ist, äußerst komplexe und unterschiedliche Daten zu speichern und dennoch auf einfache Weise analog zueinander zu verarbeiten. Um diese Prozesse zu simulieren, mussten Programme dazu gebracht werden, mit extremer Komplexität umzugehen und diese Komplexität gleichzeitig zu verbergen – so entstand einfachere, allgemeinere und verständlichere Logik mit anpassbarer, wiederverwendbarer Funktionalität. Diese Charakteristika erinnern an die Forderungen für zeitgemäße Applikationsentwicklung, und dass Objekttechnologie Entwickler in einer natürlicheren Weise arbeiten lässt, die eher dem menschlichen Denken entspricht, ist dafür ein großer Vorteil.

### OBJEKT- ODER RELATIONALER ZUGRIFF

In der Objekttechnologie sind die Daten in ihrer ganzen Komplexität im Objekt enthalten und der Zugriff darauf erfolgt über ein einfaches, einheitliches Interface. Relationale Technologie verfügt zwar ebenfalls über ein einfaches, einheitliches Interface, bietet aber keinerlei Unterstützung für die komplexen Daten der realen Welt – sie verteilt sie vielmehr über unzählige simple Tabellen. So tragen am Ende Anwender oder Entwickler die Verantwortung für den konsistenten Umgang mit den Daten.

Weil Objekte komplexe Daten einfach abbilden können, ist Objektprogrammierung am besten für komplexe Applikationen geeignet. Ebenso ist der Objektzugriff auf die Datenbank am besten geeignet, um Daten hinzuzufügen oder zu aktualisieren (also für die Transaktionsverarbeitung).

Caché ergänzt den Objektzugriff um ein objektorientiertes SQL. Diese mächtige Datenbankabfragesprache wird von den meisten Auswertungswerkzeugen unterstützt. SQL eignet sich am besten für Abfragen und Auswertungen und nicht so sehr für die Transaktionsverarbeitung (hier ist es schwerfällig und meistens ineffizient). Die Objekterweiterungen in Caché SQL eliminieren die mühevollen Join-Syntax weitgehend und erleichtern so die Verwendung von SQL ganz erheblich.

### OBJEKTMODELL UND OBJEKTPROGRAMMIERUNG VON CACHÉ IM ÜBERBLICK

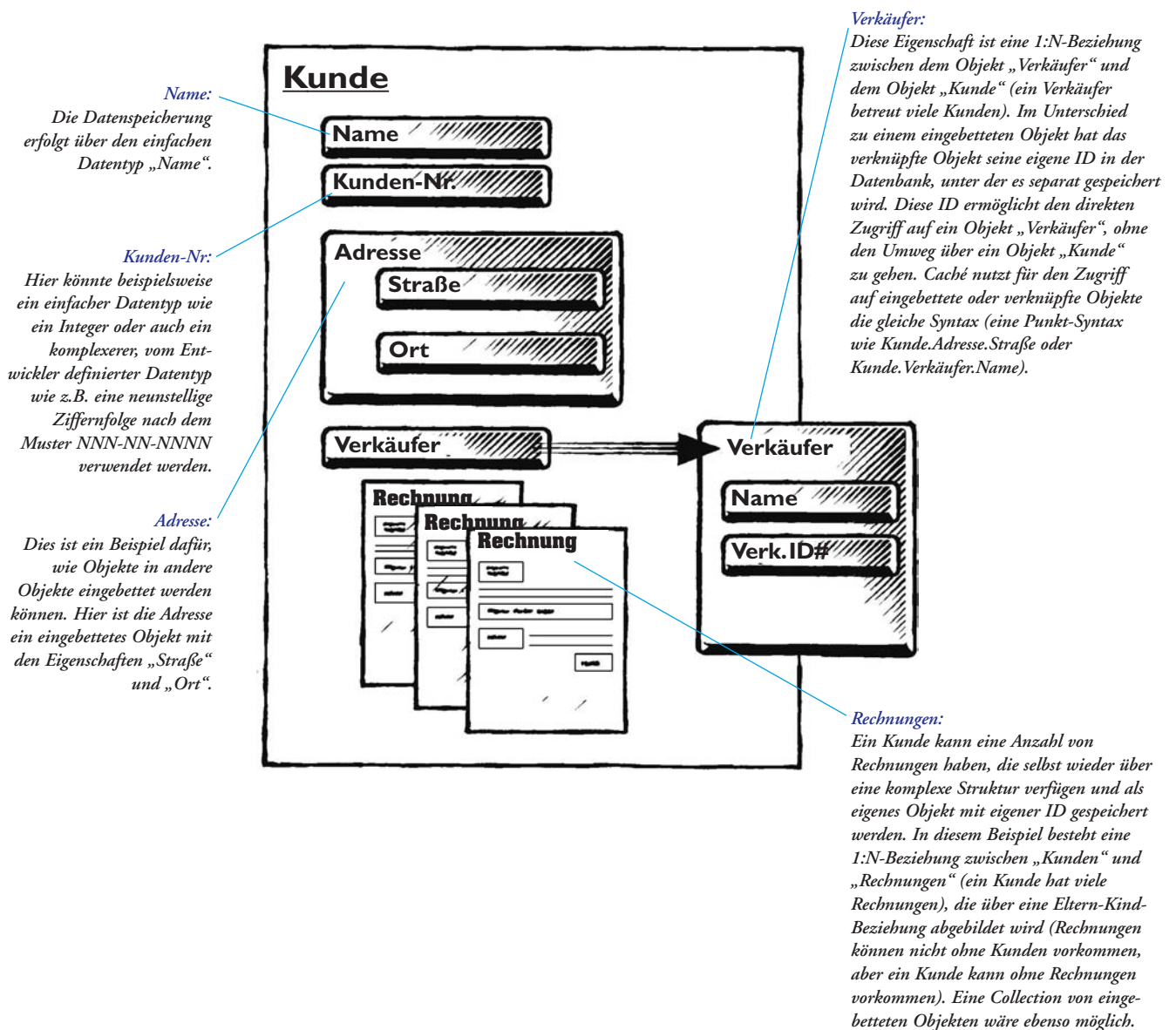
Das Objektmodell von Caché basiert auf dem Standard der ODMG (Object Database Management Group) und unterstützt zahlreiche fortschrittliche Konzepte bis hin zur Mehrfachvererbung.

Die Objekttechnologie ist der Versuch, den menschlichen Umgang mit Informationen nachzubilden. Anders als bei relationalen Tabellen bilden bei Objekten Daten und Code eine Einheit. So kann beispielsweise ein Objekt „Rechnung“ über Daten wie eine Rechnungsnummer und den Betrag sowie über eine Methode zum Drucken der Rechnung verfügen.

Ein Objekt lässt sich als ein Paket verstehen, das aus sämtlichen Datenwerten (den Eigenschaften) und einer Kopie des gesamten dazugehörigen Codes (den Methoden) des Objektes besteht. Methoden interagieren mit anderen Methoden durch das Versenden von Nachrichten. Um den Aufwand zu minimieren, nutzen Objekte der gleichen Klasse gewöhnlich einen gemeinsamen Code (es wäre ja auch unsinnig, für jedes Rechnungsobjekt tatsächlich eine Kopie der identischen Methode anzulegen). In Caché ist es zudem so, dass Methodenaufrufe intern wenn möglich als Funktionsaufrufe ausgeführt werden, um so den Aufwand eines Nachrichtenaustauschs zu vermeiden. Diese Implementierungstechniken bleiben jedoch vor dem Programmierer stets verborgen; es ist also völlig korrekt, sich Methodenaufrufe grundsätzlich als das Senden einer Nachricht vorzustellen.

Worin besteht nun der Unterschied zwischen einem Objekt und einer Klasse? In einer Klasse definiert der Entwickler die Struktur und den Code von Objekten. Klassen enthalten die allgemeine Beschreibung der Datenart (ihren „Typ“) und ihrer Speicherung sowie den gesamten Code, nicht aber die Daten selbst. Ein Objekt ist ein konkretes Exemplar („Instanz“) einer Klasse. So ist beispielsweise die Rechnung Nr. 123456 ein Objekt der Klasse „Rechnung“.

Die Objekttechnologie fördert eine natürliche Sicht auf Daten, indem sie Eigenschaften nicht auf einfache, computerzentrierte Datentypen beschränkt. Objekte können beispielsweise andere Objekte oder Referenzen auf weitere Objekte enthalten, was einer funktionellen, sinnvollen Datenmodellierung sehr entgegen kommt. Als einfaches Beispiel dafür soll ein Objekt „Kunde“ dienen:



## SCHLÜSSELBEGRIFFE DER OBJEKTTECHNOLOGIE

**Vererbung** ist die Möglichkeit, eine Klasse von Objekten von einer anderen abzuleiten. Die abgeleitete neue Klasse (eine Unterklasse) verfügt über alle Eigenschaften und Methoden ihrer Ursprungs Klasse, sie kann darüber hinaus aber auch zusätzliche eigene Eigenschaften und Methoden enthalten. Objekte einer Unterklasse haben eine „ist ein(e)“-Beziehung zu ihrer Oberklasse. Weil z. B. ein Hund ein Säugetier ist, ist es sinnvoll, die Klasse „Hund“ als Unterklasse der Klasse „Säugetier“ zu definieren, so dass sie alle Eigenschaften und Methoden von dieser erbt. Ein Beispiel für eine zusätzliche Eigenschaft in der Unterklasse wäre hier etwa die Hundemarkennummer. In einer Unterklasse können außerdem ererbte Definitionen überschrieben werden – so kann sich beispielsweise die Methode „Drucken“ einer Unterklasse von „Rechnungen“ von der Methode „Drucken“ der Oberklasse „Rechnungen“ unterscheiden. Vererbung sorgt für die Wiederverwendung von Code und macht es leichter, durchgängige Verbesserungen einzuführen.

**Mehrfachvererbung** bedeutet, dass eine Unterklasse von mehr als nur einer Oberklasse abgeleitet sein kann. Ein Hund ist beispielsweise zugleich ein Säugetier und ein Haustier. Unsere Klasse „Hund“ kann demnach die Attribute der beiden Oberklassen „Säugetier“ und „Haustier“ erben.

Unter **Kapselung** versteht man, dass Applikationen Objekte als eine Art „Black Box“ betrachten können. Öffentliche Eigenschaften und Methoden einer Klasse sind allgemein zugänglich, private Eigenschaften und Methoden stehen dagegen nur innerhalb der eigenen Objektklasse zur Verfügung. Eine Anwendung muss nicht wissen, wie ein Objekt intern funktioniert – es genügt, die öffentlichen Eigenschaften und Methoden zu verwenden. Durch die Kapselung können Programmierer die internen Abläufe in Objekten verbessern, ohne dass der Rest der Anwendung davon betroffen ist.

Als **Polymorphismus** bezeichnet man die Tatsache, dass Methoden unterschiedlicher Klassen ein gemeinsames Interface besitzen können, auch wenn die zugrunde liegende Implementierung ganz verschieden ist. Betrachten wir beispielsweise eine Applikation, die verschiedene Objektklassen wie „Brief“, „Adressaufkleber“ und „Namensschild“ nutzt. Alle drei Klassen beinhalten vielleicht eine Methode „Drucken“. Die Applikation selbst benötigt nun keine speziellen Informationen, mit welchem Objekt sie es zu tun hat – sie kann einfach die Methode „Drucken“ des gerade aktuellen Objekts aufrufen.

### VORTEILE VON CACHÉ

Caché ist vollständig objektorientiert und erlaubt es, Objekttechnologie auch für die Entwicklung hochleistungsfähiger TP-Anwendungen zu nutzen.

**Intuitive Datenmodellierung:** Dank der Objekttechnologie können Entwickler mit Informationen – auch der komplexesten Art – in einfacher und realitätsnaher Weise umgehen und so den Prozess der Anwendungsentwicklung beschleunigen.

**Schnelle Anwendungsentwicklung:** Objektorientierte Konzepte wie Kapselung, Vererbung und Polymorphismus ermöglichen, Klassen wiederzuverwenden, den Verwendungszweck von Klassen neu zu definieren und Klassen in verschiedenen Anwendungen einzusetzen, so dass die Entwickler ihre Arbeit über viele Projekte effizient nutzen können.

## WAS SPRICHT FÜR OBJEKTE ZUR DATENMODELLIERUNG?

Geht es um neue Datenbankapplikationen, entscheiden sich Entwickler aus zwei Gründen für Objekttechnologie: Komplexe Anwendungen lassen sich damit schneller entwickeln und in der Folge leichter pflegen. Denn Objekttechnologie bietet viele Vorteile:

- Objekte unterstützen komplexere Datenstrukturen, die zur Abbildung von Daten aus der realen Welt besser geeignet sind.
- Die Programmierung fällt leichter, weil der Entwicklungsprozess und die Datenmanipulation einfacher zu verfolgen sind.
- Standardklassen lassen sich problemlos durch individuelle Klassenversionen ersetzen. Dies erleichtert die Entwicklung individueller Applikationen genauso wie spätere Release-Wechsel.
- Die Kapselung erlaubt es Programmierern, die internen Abläufe einzelner Objekte zu optimieren, ohne den Rest der Anwendung zu beeinträchtigen.
- Objekte fördern auf einfache Weise die Integration unterschiedlicher Technologien und Applikationen.
- Objekttechnologie harmoniert bestens mit GUI-Oberflächen.
- Zahlreiche neue Entwicklungswerkzeuge setzen Objekttechnologie voraus.
- Objekte gewährleisten eine saubere Trennung zwischen der Benutzeroberfläche und der eigentlichen Applikation. So lässt sich selbst bei der Einführung einer neuen Front-End-Technologie (sei es das Web oder eine zukünftige Technologie) der größte Teil des Codes weiterverwenden.

## SPEICHERUNG VON OBJEKTEN...

Obwohl immer mehr Applikationen mit objektorientierten Programmiersprachen entwickelt werden, wird immer noch viel zu häufig der Versuch unternommen, Objektdaten in flache, relationale Tabellen zu zwingen. Dabei gehen die wichtigsten Vorteile der Objekttechnologie verloren.

Caché ist mit seinem multidimensionalen Datenmodell in besonderer Weise zur Speicherung komplexer Objektdaten geeignet. Das Resultat: Schnellerer Datenzugriff und schnellere Programmierung.

## ...PLUS RELATIONALER ZUGRIFF

Natürlich nutzen viele Werkzeuge (z.B. Listengeneratoren) für den Datenzugriff SQL und keine Objekttechnologie. Ein Alleinstellungsmerkmal von Caché ist, dass für jede im System definierte Objektklasse automatisch auch die volle Unterstützung für den SQL-Zugriff besteht. SQL-Werkzeuge können deshalb ohne zusätzlichen Aufwand sofort mit Caché-Daten arbeiten und profitieren zusätzlich von der außergewöhnlichen Performance des multidimensionalen Datenservers von Caché.

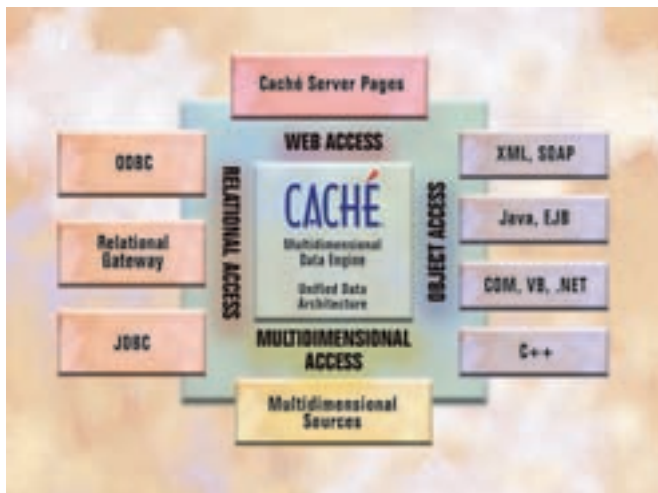
Der umgekehrte Weg ist ebenfalls möglich: Nach dem Import der DDL-Definitionen einer relationalen Datenbank generiert Caché automatisch eine passende Klassenhierarchie, so dass auch hier Objekt- und SQL-Zugriff gleichermaßen möglich werden.

Die Unified Data Architecture von Caché hält beide Repräsentationen synchron – es gibt nur eine einzige Datenbeschreibung, um die der Entwickler sich kümmern muss.

## Kapitel 2: Der multidimensionale Caché-Datenserver

Die Hochleistungsdatenbank von Caché verwendet eine multidimensionale Daten-Engine zur effizienten und kompakten Speicherung von komplex strukturierten Daten. Alle Klassen und Tabellen sind in einem Unified Data Dictionary definiert, das die Objekt- und SQL-Sicht jeweils auf die multidimensionalen Strukturen abbildet – eine Zuordnung, die Caché automatisch generieren kann.

### Multidimensionaler Zugriff



### INTEGRIERTER DATENBANKZUGRIFF

Caché lässt dem Programmierer die Wahl, ob er auf die Daten als Objekte, über SQL oder direkt als multidimensionale Strukturen zugreifen möchte. Unabhängig von der Zugriffsmethode werden alle Daten in der Datenbank von Caché immer in multidimensionalen Arrays abgelegt.

Sind die Daten einmal gespeichert, können alle drei Zugriffsmethoden gleichzeitig für dieselben Daten bei uneingeschränktem gleichzeitigem Zugriff eingesetzt werden.

Caché zeichnet sich besonders durch seine einzigartige Unified Data Architecture aus. Bei der Definition einer

Objektklasse in der Datenbank erzeugt Caché automatisch eine mit SQL nutzbare relationale Beschreibung der Daten. Ebenso erzeugt Caché automatisch nicht nur eine relationale sondern auch eine Objektbeschreibung der Daten, wenn DDL-Definitionen einer relationalen Datenbank in das Data Dictionary importiert werden, so dass der Zugriff auf Objekte sofort möglich ist. Dabei stellt Caché sicher, dass diese Beschreibungen koordiniert bleiben und nur eine einzige Datendefinition gepflegt werden muss. Der Programmierer kann alle Daten und auch das Dictionary selbst sowohl aus Objektsicht als auch als relationale Tabellen bearbeiten und anzeigen.

Caché erstellt automatisch eine Zuordnung, die festlegt, wie Objekte und Tabellen in den multidimensionalen Strukturen gespeichert werden. Alternativ kann der Programmierer diese Zuordnung allerdings auch explizit festlegen.

### VORTEILE VON CACHÉ

**Flexibilität:** Die Datenzugriffsmodi von Caché – Objekte, SQL und multidimensional – können gleichzeitig für dieselben Daten verwendet werden. Diese Flexibilität erlaubt es dem Programmierer, Daten immer auf die Weise zu betrachten, die gerade am geeignetsten ist, und stets die Zugriffsmethode zu verwenden, die am besten zu den Anforderungen des entsprechenden Programms passt.

**Weniger Aufwand:** Die Unified Data Architecture von Caché beschreibt die Daten in einer einzigen Definition als Objekte und als Tabellen. Transformationen werden überflüssig, so dass Anwendungen einfacher entwickelt und gepflegt werden können.

**Vorhandene Kenntnisse und Anwendungen nutzen:** Programmierer können ihre relationalen Kenntnisse nutzen und bei der Weiterentwicklung vorhandener Anwendungen nach und nach objektorientierte Elemente einsetzen.

## DAS MULTIDIMENSIONALE DATENMODELL

Die Caché-Datenbank basiert auf einer extrem leistungsstarken multidimensionalen Daten-Engine. Der direkte Zugriff auf multidimensionale Strukturen geschieht über die integrierten Skriptsprachen von Caché, die für höchste Performance und umfassende Speichermöglichkeiten sorgen. Viele Anwendungen sind sogar vollständig in der Daten-Engine von Caché implementiert. Der direkte "Global-Zugriff" wird häufig bei unüblichen oder sehr speziellen Strukturen verwendet, wenn der objektorientierte oder SQL-basierte Zugriff nicht erforderlich ist, oder wenn die höchstmögliche Performance gefordert ist.

Die multidimensionale Daten-Engine selbst kennt kein Data Dictionary und folglich auch keine Datendefinitionen.

### Komplexe multidimensionale Datenstrukturen

Die multidimensionalen Arrays von Caché werden als „Globals“ bezeichnet. In einem Global können Daten mit einer beliebigen Anzahl von Subscripts gespeichert werden. Darüber hinaus sind die Subscripts nicht an bestimmte Typen gebunden: String, Integer, Fließkomma – alles ist möglich. Das heißt, ein Subscript kann ein Integer-Wert wie „34“ sein, ein anderes eine Bezeichnung wie „RechnungsPosition“ – und das alles auf der gleichen Subscript-Ebene.

In einer Inventur-Anwendung, die Informationen über Artikel und deren Größe, Farbe und Muster enthält, könnte die Datenstruktur beispielsweise so aussehen:

```
^Bestand(Artikel,Größe,Farbe,Muster) = Lagerbestand
```

Dazu ein Datenbeispiel:

```
^Bestand("Kleid",38,"blau","Blütenmuster")=3
```

Mit dieser Datenstruktur lässt sich problemlos feststellen, ob Kleider der Größe 38 in blau mit Blütenmuster auf Lager sind – ganz einfach durch Zugriff auf genau diesen Datenknoten. Möchte eine Kundin nun zwar ein Kleid der Größe 38, ist sich aber über Farbe und Muster noch unschlüssig, dann wird einfach eine Liste aller Kleider der Größe 38 erstellt, indem alle Datenknoten unterhalb der Ebene des folgenden Datenknotens durchsucht werden:

```
^Bestand("Kleid",38)
```

In diesem ersten Beispiel hatten alle Datenknoten einen ähnlichen Aufbau (sie enthielten den Lagerbestand) und wurden alle auf der gleichen Subscript-Ebene (vier Subscripts) mit ähnlichen Subscripts (das dritte Subscript war immer eine Textbezeichnung für die Farbe) gespeichert. Das muss jedoch keineswegs immer so sein: Datenknoten müssen weder über die gleiche Anzahl noch über die gleichen Typen von Subscripts verfügen, und sie können durchaus ganz unterschiedliche Datentypen enthalten.

Hier deshalb ein Beispiel für einen etwas komplexeren Global mit Rechnungsdaten, bestehend aus unterschiedlichen Datentypen, die auf verschiedenen Subscript-Ebenen gespeichert sind:

```
^Rechnung(Rechnungsnummer,"Datum") = Rechnungsdatum
```

```
^Rechnung(Rechnungsnummer,"Kunde") = Kundeninformation
```

```
^Rechnung(Rechnungsnummer,"Positionen") = Anzahl der Positionen der Rechnung
```

```
^Rechnung(Rechnungsnummer,"Positionen",1,"Artikel") = Artikelnummer der 1.Position
```

```
^Rechnung(Rechnungsnummer,"Positionen",1,"Menge") = Menge der ersten Position
```

```
^Rechnung(Rechnungsnummer,"Positionen",1,"Preis") = Preis der ersten Position
```

```
^Rechnung(Rechnungsnummer,"Positionen",2,"Artikel") = Artikelnummer der 2. Position
```

**usw.**

### Mehrere Datenelemente pro Knoten

Oftmals enthält ein Datenknoten lediglich ein Datenelement, z. B. ein Datum oder eine Stückzahl. Gelegentlich ist es aber auch zweckmäßig, mehrere Datenelemente gemeinsam in einem einzigen Knoten zu speichern. Vor allem ist dies dann nützlich, wenn es sich um Daten handelt, die in enger Beziehung zueinander stehen und deshalb oft gemeinsam aufgerufen werden. Dies kann auch zur Performance-Verbesserung beitragen, weil weniger häufig auf die Datenbank zugegriffen werden muss.

In unserem vorherigen Rechnungsbeispiel enthielten die Rechnungspositionen eine Artikelnummer mit Menge und Preis, wobei jede Position als separater Knoten gespeichert wurde. Alternativ könnte man die Angaben für jede Position jeweils zusammen in einem Knoten speichern:

```
^Rechnung(Rechnungsnummer,"Positionen",Positionsnummer)
```

Mit Caché geht das recht einfach, weil die Funktion „\$List“ mehrere Datenelemente in einen Binärstring packen und später unter Beibehaltung der unterschiedlichen Datentypen wieder entpacken kann. Elemente können wiederum Unterelemente enthalten, diese wiederum ebenfalls, etc.

### Logische Sperren: Konfliktvermeidung bei konkurrierenden Prozessen

Die Minimierung von Konflikten zwischen konkurrierenden Prozessen ist eine der Grundvoraussetzungen für höchste Performance in Systemen mit Tausenden von Benutzern. Einer der Hauptkonflikte besteht dabei zwischen Transaktionen, die Zugriff auf die gleichen Daten erfordern.

In Caché sperren Transaktionen beim Update von Daten keine kompletten physikalischen Seiten. Weil häufig nur Zugriffe oder Änderungen kleinster Datenmengen notwendig sind, erfolgen alle Datenbanksperren stattdessen auf logischer Ebene. Zusätzlich werden Datenbankkonflikte durch atomare Additions- und Subtraktionsoperationen reduziert, die überhaupt keine Sperren benötigen. (Solche Operationen sind besonders hilfreich beim Heraufsetzen von Kennungen oder für statistische Zähler.)

So laufen mit Caché die einzelnen Transaktionen schneller und es können wesentlich mehr Transaktionen gleichzeitig ausgeführt werden.

### Sparse Arrays mit variablen Datenlängen

Kennzeichnend für Caché sind variable Datenlängen und deren Speicherung in Sparse Arrays. Im Vergleich zu relationalen Datenbanken sinkt deshalb der Speicherbedarf häufig um mehr als 50%. Aber die kompakte Datenspeicherung macht sich nicht nur in den geringeren Anforderungen an die Plattenkapazitäten bemerkbar: Weil mit einer einzelnen Operation mehr Daten gelesen oder geschrieben werden können und die Daten auch effizienter im Cache abgelegt werden, wirkt sich dies auch noch zusätzlich positiv auf die Performance aus.

### Deklarationen und Definitionen sind überflüssig

Die multidimensionalen Arrays von Caché sind grundsätzlich typpfrei, sowohl was die Daten als auch was die Subscripts angeht. Es sind weder Deklarationen, Definitionen noch Speicherzuweisungen erforderlich. Global-Daten entstehen ganz einfach, wenn sie erstmals in der Datenbank gespeichert werden.

### Namespaces

In Caché werden Daten und Code in Dateien mit der Bezeichnung CACHE.DAT gespeichert (jeweils eine Datei pro Verzeichnis). Jede dieser Dateien enthält viele Globals (multidimensionale Arrays). Innerhalb einer Datei muss jeder Global-Name eindeutig sein, aber unterschiedliche Dateien können Globals mit gleichen Namen enthalten. Diese Dateien stellen, stark vereinfacht betrachtet, die Datenbanken dar.



Um auf Daten zuzugreifen, müssen Caché-Prozesse allerdings nicht wissen, wo genau sich die betreffende CACHE.DAT-Datei befindet. Sie verwenden zur Adressierung vielmehr einen „Namespace“. Ein solcher Namespace ist eine logische Zuordnung der Namen der multidimensionalen Globals und des Codes zu den einzelnen Datenbanken. Wird eine Datenbank auf eine andere Festplatte oder einen anderen Rechner verschoben, muss nur die Namespace-Zuordnung aktualisiert werden – die Anwendung selbst bleibt unverändert.

In der Regel werden alle zu einem Namespace gehörenden Daten, außer einiger Systeminformationen, in einer einzigen Datenbank gespeichert. Namespaces stellen jedoch eine flexible Struktur dar, die eine beliebige Zuordnung erlaubt. So können die Inhalte eines Namespaces durchaus aus verschiedenen Datenbanken zusammengestellt werden, wobei sich diese sogar auf verschiedenen Rechnern befinden können.

## **VORTEILE VON CACHE**

**Performance:** Der Einsatz eines multidimensionalen Datenmodells mit Sparse-Array-Speichertechnik ermöglicht im Vergleich zum unhandlichen Gewirr zweidimensionaler Tabellen Datenzugriff und Aktualisierungen mit erheblich reduzierten Festplatten-Zugriffen. Für Anwendungen heißt das: Sie laufen ganz einfach schneller.

**Skalierbarkeit:** Caché-basierte Anwendungen sind aufgrund ihres transaktionsorientierten multidimensionalen Datenmodells auf viele Tausende von Clients skalierbar, ohne die hohe Performance einzubüßen. Das liegt daran, dass der Datenzugriff im multidimensionalen Modell im Gegensatz zum relationalen Modell nur unwesentlich von Größe oder Komplexität der Datenbank beeinflusst wird. Transaktionen greifen ohne komplizierte Join-Operationen oder das berüchtigte „Table-Hopping“ direkt auf die benötigten Daten zu.

Ein wichtiger Faktor für das reibungslose Funktionieren der gleichzeitigen Zugriffe sind die logischen Sperren in Caché – anstatt der Sperre ganzer physikalischer Seiten – wie auch das ausgefeilte Daten-Caching über Netzwerke.

**Schnelle Anwendungsentwicklung:** Caché beschleunigt die Anwendungsentwicklung, weil die Datenstruktur die natürliche, einfach verständliche Speicherung komplexer Daten ermöglicht und umfangreiche oder komplexe Deklarationen und Definitionen überflüssig werden. Der direkte Zugriff auf Globals ist sehr einfach und erlaubt die Verwendung der gleichen Sprachsyntax wie beim Zugriff auf lokale Arrays.

**Wirtschaftlichkeit:** Im Vergleich zu relationalen Anwendungen ähnlicher Größe kommen Caché-basierte Anwendungen mit erheblich weniger Hardware und ohne Datenbankadministratoren aus. Systemverwaltung und laufender Betrieb sind ausgesprochen einfach.

## SQL-ZUGRIFF

Caché verwendet als Abfragesprache SQL, wobei der volle Umfang relationaler Datenbankfunktionalität unterstützt wird – einschließlich DDL, Transaktionen, referenzieller Integrität, Triggern, Stored Procedures und vielem mehr. Caché unterstützt den Datenzugriff über ODBC und JDBC (mittels eines reinen Java-basierten Treibers). SQL-Befehle und Abfragen können darüber hinaus auch in Caché ObjectScript und in Objektmethoden eingebettet werden.

Für den Zugriff mit SQL werden Daten als Tabellen mit Zeilen und Spalten betrachtet. Da in Caché intern jedoch alle Daten in effizienten multidimensionalen Strukturen gespeichert sind, erzielen Anwendungen mit Caché eine höhere Performance als mit traditionellen relationalen Datenbanken, auch wenn sie SQL einsetzen.

Neben der SQL-Standardsyntax unterstützt Caché eine Vielzahl der in anderen Datenbanken häufig verwendeten Erweiterungen, so dass viele SQL-basierte Anwendungen ohne Änderung auf Caché laufen können – dies gilt vor allem für diejenigen, die mit datenbankunabhängigen Werkzeugen erstellt wurden. Proprietäre Stored Procedures erfordern allerdings etwas Nachbearbeitung, hier kann InterSystems den Umstieg mit entsprechenden Migrationswerkzeugen erleichtern.

Caché SQL zeichnet sich durch objektorientierte Erweiterungen aus, die den SQL-Code vereinfachen und wesentlich intuitiver gestalten.

### NORMALES SQL

```

SELECT
VK.Name, VM.Beschr, HV.Wert,
VP.InvDatum, VP.InvNummer
FROM
HauptVerkauf HV, VerkaufPosition VP,
VerkaufProdukt VPR, VerkaufKunde VK,
VerkaufMarkt VM
WHERE
VP.VerkaufsPositionID *=
HV.VerkaufPosition
AND VPR.VerkaufProduktID *= HV.Produkt
AND VK.VerkaufKundeID *= HV.Kunde
AND VM.VerkaufMarktID *= VK.VerkaufMarkt
AND VPR.Beschr = 'Hammer'

```

### SQL MIT OBJEKTORIENTIERTEN

#### ERWEITERUNGEN

```

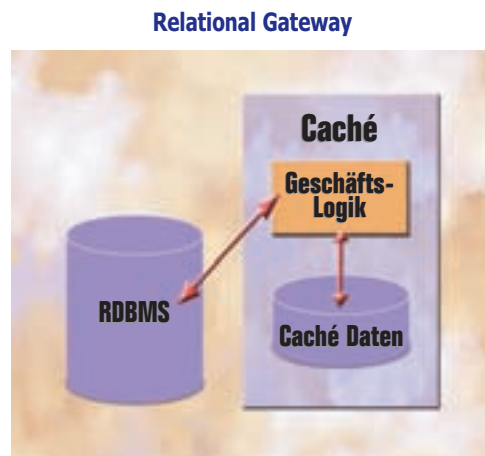
SELECT
Kunde->Name,
Kunde->VerkaufMarkt->Beschr, Wert,
VerkaufPosition->InvDatum,
VerkaufPosition->InvNummer
FROM HauptVerkauf
WHERE Produkt->Beschr = 'Hammer'

```

## Zugriff auf relationale Datenbanken mit dem Caché Relational Gateway

Über das Caché Relational Gateway können in Caché erzeugte SQL-Befehle zur Ausführung an andere (relationale) Datenbanken geschickt werden. Eine Caché-Anwendung ist damit in der Lage, auch Daten aus bestehenden relationalen Datenbanken zu lesen und zu verarbeiten.

Wenn die Caché-Datenklassen mit der Option „CachéSQLStorage“ kompiliert werden, können Caché-Anwendungen über das Gateway sogar direkt mit relationalen Datenbanken arbeiten. Eine höhere Performance und Skalierbarkeit erreichen Applikationen allerdings auf Basis von Caché.



### VORTEILE VON CACHE

**Schnelleres SQL:** Relationale Anwendungen werden signifikant schneller, wenn sie über Caché SQL auf Caché als effiziente postrelationale Datenbank aufsetzen.

**Schnellere Entwicklung:** In Caché können SQL-Abfragen intuitiver und mit weniger Code-Zeilen erstellt werden.

**Kompatibilität mit relationalen Anwendungen und Auswertungswerkzeugen:** Die nativen ODBC- und JDBC-Treiber von Caché ermöglichen relationalen Anwendungen und Auswertungswerkzeugen einen hochperformanten Zugriff auf die Caché-Datenbank. Mit dem Caché Relational Gateway können Caché-Anwendungen über SQL auf andere (relationale) Datenbanken zugreifen.

## OBJEKTE IN CACHE

Das Objektmodell von Caché basiert auf dem ODMG-Standard und unterstützt die ganze Palette an objektorientierten Programmierkonzepten wie Kapselung, eingebettete Objekte, (Mehrfach-) Vererbung, Polymorphismus und Collections.

Mit den in Caché integrierten Skriptsprachen können Objekte direkt bearbeitet werden. Caché stellt Caché-Klassen auch als Java-, EJB-, COM-, .NET- und C++-Klassen dar. In Caché Studio können Klassen außerdem durch einfachen Mausklick automatisch XML- und SOAP-fähig gemacht werden. Somit stehen die Caché-Objekte zum Einsatz für jede gängige Objekttechnologie ohne weiteres zur Verfügung.

Für Programme außerhalb des Caché-Applikationsservers bestehen verschiedenen Möglichkeiten, auf Caché-Klassen zuzugreifen:

1. Caché-Klassen können als native Klassen für verschiedene Sprachen projiziert werden. Wenn ein Java-, C++, C#- oder anderes Programm auf ein Caché-Objekt zugreift, ruft dieses eine Projektion der Klasse in der betreffenden Sprache auf. Diese projizierte Klasse, die von Caché automatisch erzeugt wird, kommuniziert mit dem Caché-Applikationsserver, um Methoden auf dem Caché-Server aufzurufen und auf Eigenschaften zuzugreifen oder diese zu ändern. Der Zustand von Caché-Objekten wird immer im Caché-Applikationsserver verwaltet. Um die Ausführung zu beschleunigen und den Nachrichtenaustausch zu verringern, legt Caché eine Kopie der Daten des Objekts im Cache des Clients ab und überträgt Aktualisierungen nach Möglichkeit im Huckepackverfahren mit anderen Nachrichten.
2. In einer "abgespeckten" Variante der Projektion greift die projizierte Klasse in der jeweiligen Sprache direkt auf die Datenbank zu und umgeht den Caché-Applikationsserver. Hier wird der Zustand des Objekts nicht auf dem Applikationsserver gehalten und die Eigenschaften im Speicher werden nur auf dem Client verwaltet. Dieser Ansatz zeichnet sich durch einen signifikant höheren Durchsatz aus, bietet aber eine geringere Funktionalität, da serverseitige Instanzmethoden der Klasse (d.h. Methoden, die Zugriff auf die Eigenschaften im Speicher benötigen) nicht aufgerufen werden können.
3. Die Jalapeño-Technologie von InterSystems ist speziell für Java-Entwickler gedacht. Sie können damit zuerst in ihrer bevorzugten Java-Entwicklungsumgebung die Java-Datenbankklassen wie jede andere Java-Klasse erstellen. Auf Knopfdruck übernimmt Caché dann die Generierung eines Datenbankschemas und der entsprechenden Caché-Klassen daraus. Bei diesem Ansatz bleibt die Java-Klasse unverändert, und die Anwendung greift weiterhin auf deren Eigenschaften und Methoden zu. Caché bietet eine Bibliotheksklasse (den „ObjectManager“) mit einem API, das zum Speichern und Abrufen von Datenbankobjekten und für Abfragen verwendet wird.

In jedem dieser drei Ansätze erscheinen Objekte für das Benutzerprogramm als lokal. Caché verwaltet auf transparente Weise die gesamte Kommunikation, entweder per Call-in oder über TCP.

Die Java Vorlagen- und Hilfs-Bibliotheken basieren vollständig auf Java, so dass sie über das Web genutzt oder auch auf speziellen Java-Geräten eingesetzt werden können.

## Methodengeneratoren

Caché zeichnet sich durch eine Reihe modernster Objekttechnologien aus – eine davon sind die Methodengeneratoren. Ein Methodengenerator ist eine Methode, die bei der Kompilierung ausgeführt wird und ihrerseits wieder Code erzeugt, der dann bei der späteren Ausführung des Programms abläuft. Ein Methodengenerator hat Zugriff auf Klassendefinitionen, einschließlich der Eigenschafts- und Methodendefinitionen und -parameter, damit er eine für die Klasse angepasste Methode erzeugen kann. Methodengeneratoren sind besonders leistungstark in Zusammenhang mit der Mehrfachvererbung – damit kann eine Funktionalität aus einer zusätzlichen Oberklasse geerbt werden und sich selbst an die Unterklasse anpassen.



## VORTEILE VON CACHÉ

Caché ist vollständig objektorientiert und erlaubt den Einsatz der ganzen Bandbreite von Objekttechnologie für Anwendungen zur Hochleistungs-Transaktionsverarbeitung.

### Schnelle Anwendungsentwicklung:

Die Objekttechnologie ist ein leistungstarkes Werkzeug zur Erhöhung der Produktivität der Programmierer. Mit ihrer Hilfe lassen sich Objekte – so komplex sie auch sein mögen – in einfacher und realistischer Weise betrachten und damit der Anwendungsentwicklungsprozess erheblich beschleunigen. Da Objekte naturgemäß modular aufgebaut und interoperabel sind, vereinfacht dies die Pflege von Anwendungen und erlaubt Programmierern die Nutzung ihrer Arbeiten über viele Projekte hinweg.

### Natürliche Entwicklung:

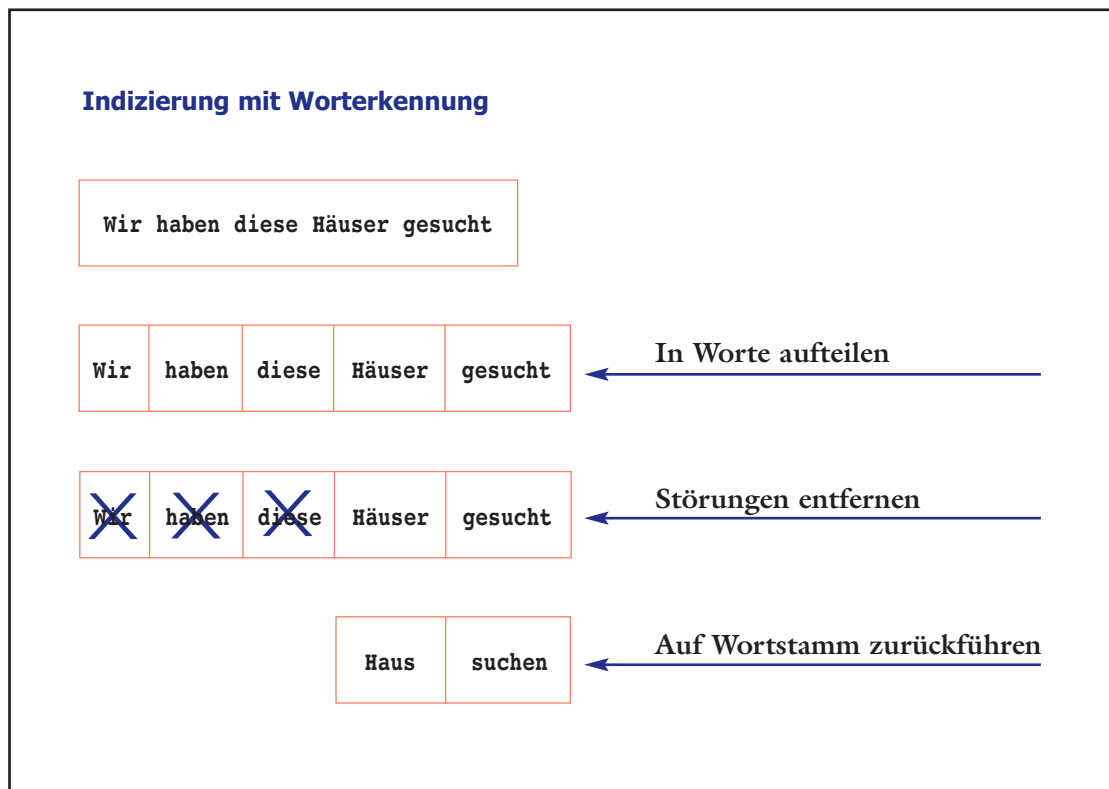
Datenbankobjekte sind in der vom Entwickler verwendeten Sprache als native Objekte verfügbar. Es ist nicht mehr notwendig, umständlich Code zur Zerlegung von Objekten in Zeilen und Spalten zu erstellen und diese dann später neu zusammensetzen.

## TEXTSUCHE MIT WORTERKENNUNG

Caché unterstützt eine komfortable Freitextsuche, bei der Abfragen Wörter im Text auch dann finden, wenn es sich um Varianten des Suchworts handelt. Um die Suche mit Worterkennung nutzen zu können, muss das Textfeld einen entsprechenden Index-Typ („Word-Aware“) besitzen. Der Algorithmus zur Erstellung eines solchen Index funktioniert wie folgt:

1. Zuerst werden die einzelnen Wörter im Textfeld identifiziert.
2. Gängige Wörter, die für eine aussagekräftige Suche nicht geeignet sind, werden entfernt, z. B. „der“, „die“, „das“ oder „für“.
3. Die restlichen Wörter werden auf ihre Wortstämme reduziert, z.B. „suchend“ wird zu „suchen“ und „Blumen“ zu „Blume“.
4. Die daraus resultierenden Wörter werden indiziert.

In der Suche mit Worterkennung wird der Suchtext in der Regel zuerst auf ähnliche Weise verarbeitet, und dann werden mit dem Index Übereinstimmungen (Matches) erzeugt.



Indizes mit Worterkennung werden von Objekt- und SQL-Updates geführt. Die Suche erfolgt meist über SQL-Abfragen, obwohl auch mit prozeduralem Code direkt auf die Indizes zugegriffen werden kann. Solche Abfragen können AND/OR-Logik für ausgefeiltere Suchvorgänge enthalten. Die Algorithmen für die Worterkennung sind speziell auf die verwendete natürliche Sprache abgestimmt. Die Suche mit Worterkennung ist für viele natürliche Sprachen verfügbar, wie Deutsch, Englisch, Französisch, Italienisch, Portugiesisch und Spanisch. Weitere werden hinzukommen.

### Suche mit Worterkennung

**WHERE Beschreibung**

**%Contains ('Haus')**

**Caché findet "Haus",  
"Häuser", ...**

**Nicht aber "zu Hause" oder  
"Hausbau".**

### VORTEILE VON CACHÉ

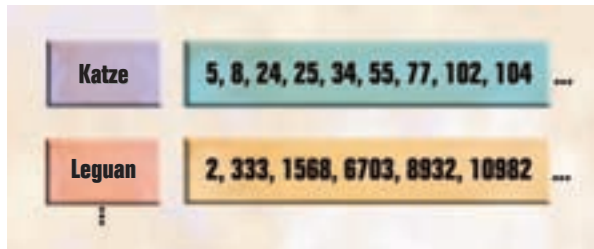
**Leistungsstarke unstrukturierte Textsuche:** Unstrukturierter Text, wie ärztliche Aufzeichnungen oder Dokumente, können einfach nach Schlüsselwörtern und verwandten Wörtern durchsucht werden.

**Extrem schnelle Suche:** Wenn man die Worterkennung mit der transaktionalen Bitmap-Indizierung von Caché verbindet, können extrem umfangreiche Textmengen in Sekundenbruchteilen durchsucht werden.

## TRANSAKTIONALE BITMAP-INDIZIERUNG

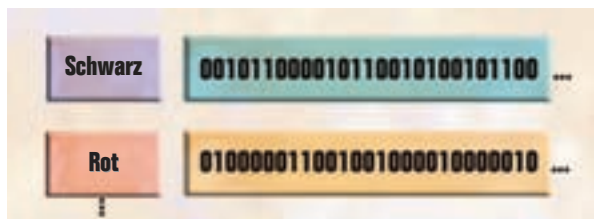
Eine einzigartige Möglichkeit von Caché stellt die transaktionale Bitmap-Indizierung dar, die die Performance komplexer Abfragen signifikant erhöhen kann und Abfragen auf Live-Daten mit einer sonst nur in Data-Warehouse-Abfragen erzielbaren Geschwindigkeit ausführt.

### Herkömmlicher Index für die Eigenschaft "Haustier"



Die Performance einer Datenbank hängt entscheidend davon ab, ob Indizes auf Eigenschaften vorhanden sind, die häufig zum Durchsuchen einer Datenbank verwendet werden. Konventionell enthält ein Index in einer Datenbank für jeden möglichen Wert der Spalte oder der Eigenschaft eine Liste mit den Kennungen derjenigen Zeilen oder Objekte, die diesen Wert aufweisen.

### Bitmap-Index für die Eigenschaft "Haarfarbe"



Ein Bitmap-Index funktioniert etwas anders. Hier gibt es für jeden möglichen Wert einer Spalte oder Eigenschaft einen Bitstring, in dem jedes Bit für eine Zeile oder ein gespeichertes Objekt steht. Ein Bitwert von 1 bedeutet, dass die Zeile oder das Objekt den entsprechenden Wert für diese Spalte oder Eigenschaft aufweist.

Der Vorteil von Bitmap-Indizes liegt nun darin, dass komplexe Abfragen durch die Anwendung von booleschen Operatoren (UND, ODER, etc.) auf die Bitstrings verarbeitet werden können – so lässt sich effizient ermitteln, welche Objekte (Zeilen) die Abfragebedingungen erfüllen, ohne dass die gesamte Datenbank durchsucht werden muss. Bitmap-Indizes beschleunigen die Antwortzeit bei Abfragen großer Datenvolumen oftmals um den Faktor 100 oder höher.

Allerdings bringen Bitmaps in der Regel zwei Nachteile mit sich: a) In relationalen Datenbanken ist die Aktualisierung oft extrem langsam. b) Sie belegen oft viel zu viel Speicher. Daher werden sie in relationalen Datenbanken selten für Anwendungen bei der Transaktionsverarbeitung eingesetzt.

Caché enthält eine neue Technologie, die so genannten transaktionalen Bitmap-Indizes, die diese beiden Nachteile mit Hilfe von multidimensionalen Datenstrukturen eliminiert. Die Aktualisierung solcher Bitmaps ist oft sogar schneller als bei herkömmlichen Indizes. Zudem wird mit ausgefeilten Komprimierungstechniken gearbeitet, um den Speicherbedarf drastisch zu senken. Caché unterstützt auch ausgefeilte „Bit-Slicing“-Techniken. Das Ergebnis sind ultraschnelle Bitmaps, mit denen bei einer Online-Datenbank oftmals Millionen von Datensätzen im Bruchteil einer Sekunde durchsucht werden können. Business Intelligence- und Data Warehousing-Anwendungen können dadurch mit „Live“-Daten, also in Echtzeit arbeiten.

Caché unterstützt sowohl herkömmliche als auch transaktionale Bitmap-Indizes, auch über mehrere Spalten. So kann man beispielsweise mit einem Index auf „Bundesland“ und „FahrzeugModell“ schnell jeden Fahrer ermitteln, der in einem bestimmten Bundesland ein bestimmtes Auto fährt.



## VORTEILE VON CACHE

**Dramatisch schnellere Abfragen:** Durch den Einsatz transaktionaler Bitmap-Index-Techniken werden große Datenbanken mit rasanter Geschwindigkeit durchsucht – häufig Millionen von Datensätzen im Bruchteil einer Sekunde –, und das auf einem System, das primär zur Transaktionsverarbeitung verwendet wird.

**Datenanalysen in Echtzeit:** Mit der transaktionalen Bitmap-Indizierung von Caché lassen sich Datenanalysen in Echtzeit auf den aktuellen Daten durchführen.

**Geringere Kosten:** Ein zweiter Rechner für das Data Warehouse oder die Entscheidungsunterstützung ist überflüssig. Ebenso entfällt auch das tägliche Übertragen von Daten auf ein zweites System und man braucht keinen Datenbankadministrator, der sich um das System kümmert.

**Skalierbarkeit:** Die hohe Geschwindigkeit der transaktionalen Bitmaps sorgt dafür, dass problemlos Systeme mit riesigen Datenmengen aufgebaut, gepflegt und regelmäßig durchsucht werden können.

## DAS ENTERPRISE CACHE PROTOCOL FÜR VERTEILTE SYSTEME

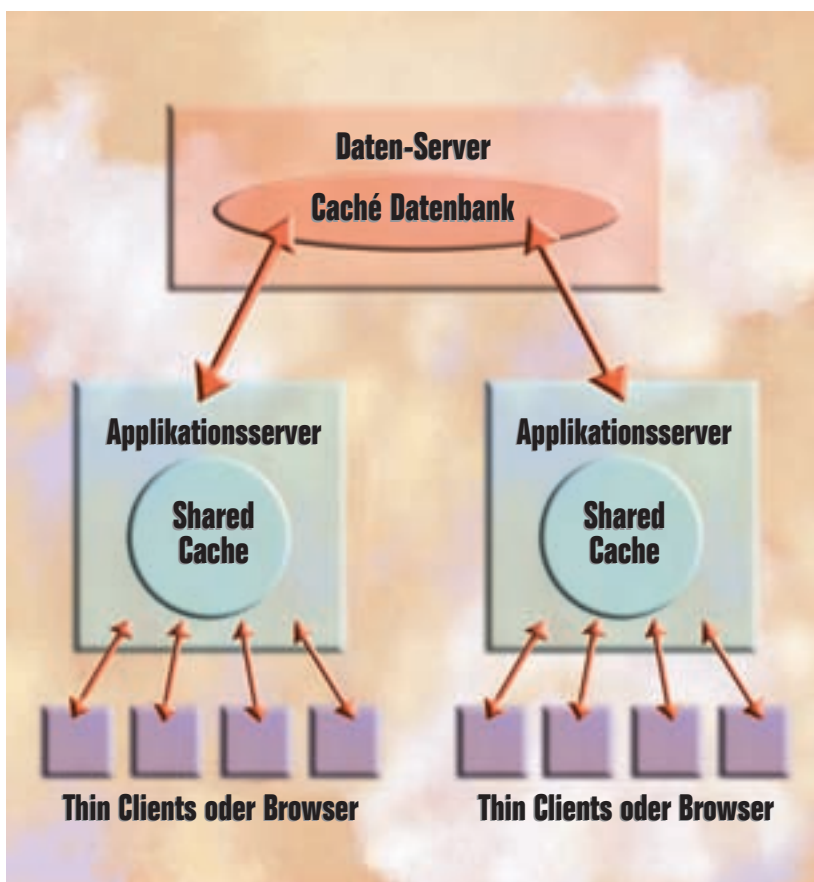
### Skalierbare Performance in verteilten Systemen

Das Enterprise Cache Protocol (ECP) von InterSystems ist eine skalierbare Hochleistungstechnologie, mit der Rechner in einer verteilten Konfiguration Datenbanken gemeinsam nutzen können. Der Einsatz von ECP erfordert keine Änderungen der Anwendungen – Applikationen behandeln die Datenbank ganz einfach, als ob sie lokal vorhanden wäre.

ECP arbeitet folgendermaßen: Jeder Caché-Applikationsserver enthält einen eigenen Datenserver, der Daten verarbeiten kann, die sich auf seinem eigenen Festplattensystem befinden oder die mit ECP von einem anderen Datenserver zu ihm übertragen wurden. Wenn ein Client Informationen anfordert, die sich auf einem entfernten Datenserver befinden, versucht der Applikationsserver die Anforderung aus dem lokalen Cache zu bedienen. Ist dies nicht möglich, fordert er die notwendigen Daten vom entfernten Datenserver an. Die Antwort enthält den Datenbankblock bzw. die Datenbankblöcke, in denen die Daten gespeichert sind. Diese Blöcke werden im Cache des Applikationsservers abgelegt, so dass sie hiermit allen

Anwendungen zur Verfügung stehen, die auf diesem Server ausgeführt werden. ECP stellt automatisch die Konsistenz des Caches über das Netzwerk sicher und leitet Änderungen an die betroffenen Datenserver weiter.

ECP bietet grundlegende Performance- und Skalierbarkeitsvorteile. Die Antwortzeiten auf den Clients sinken, weil benötigte Daten sich häufig bereits im lokalen Cache befinden. Darüber hinaus verringert Caching den Netzverkehr zwischen der Datenbank und den Applikationsservern, so dass jedes Netzwerk viel mehr Server und Clients unterstützen kann. Während die meisten Anwendungen von ECP erheblich profitieren, gibt es allerdings auch einige, deren Architektur eine solche Skalierung weniger gut unterstützt. Hier empfehlen sich Leistungstest, da ein paar einfache Änderungen oftmals bereits zu einer wesentlich höheren Performance führen können.





### **Einfach einzusetzen – keine Änderungen an den Anwendungen**

Der Einsatz von ECP ist für die Anwendungen transparent. Anwendungen, die für die Ausführung auf einem einzelnen Server geschrieben wurden, laufen ohne Änderung auch in einer Multi-Server-Umgebung. Um mit ECP zu arbeiten, bestimmt der System-Manager einfach für jeden Applikationsserver einen oder mehrere Datenserver. Mit der Namespace-Zuordnung lässt sich dann definieren, welche Global-Strukturen (oder Teile der Global-Strukturen) auf diesen entfernten Datenservern liegen sollen.

### **Flexibilität bei der Konfiguration**

Jedes Caché-System kann für andere Systeme sowohl als Applikationsserver als auch als Datenserver fungieren. ECP unterstützt jede beliebige Kombination aus Applikationsservern und Datenservern sowie jede Punkt-zu-Punkt-Topologie mit bis zu 255 Systemen.

## **VORTEILE VON CACHÉ**

**Massive Skalierbarkeit:** Das Enterprise Cache Protocol von Caché unterstützt bei steigenden Anforderungen das dynamische Hinzufügen von Applikationsservern, wobei jeder die Datenbank so verwendet, als ob diese lokal vorhanden wäre. Wird der Festplattendurchsatz zum Engpass, können weitere Datenserver hinzugefügt und die Datenbank entsprechend logisch partitioniert werden.

**Höhere Verfügbarkeit:** Da die Benutzer auf mehrere Rechner verteilt sind, betrifft der Ausfall eines Applikationsservers weniger Benutzer. Falls ein Datenserver abstürzt und neu gebootet werden muss oder wenn das Netzwerk vorübergehend ausfällt, können die Applikationsserver weiterarbeiten, ohne dass man mehr als eine kleine Pause wahrnimmt. Die Konfiguration mehrerer Datenserver als ausfallsicheres Hardware-Cluster kann die Verfügbarkeit enorm erhöhen.

**Geringere Kosten:** Viele preiswerte Rechner können zu einem extrem leistungsstarken System zusammengefügt werden, das die massiv parallele Verarbeitung unterstützt – „Grid Computing“.

**Transparente Nutzung:** Anwendungen müssen nicht speziell für ECP geschrieben sein – Caché-Anwendungen nutzen ohne Änderungen automatisch die Vorteile von ECP.

## FEHLERTOLERANZ

Selbst in den sichersten Umgebungen können unerwartete Ereignisse eintreten – Hardware- und Stromausfälle oder Katastrophen wie Flut oder andere Naturereignisse – trotzdem können es sich Krankenhäuser, Telekommunikationseinrichtungen und viele andere Unternehmen nicht leisten, einfach auszufallen. Um diesem hohen Standard gerecht zu werden, ist Caché so angelegt, dass es sich nach Ausfällen schnell wiederherstellen lässt und verschiedene Failover- und anderweitige Optionen bietet, um die Auswirkungen unerwarteter Ereignisse auf die Benutzer zu minimieren.

Caché Write-Image Journaling stellt zusammen mit anderen Funktionen die Integrität der Datenbank für die meisten Typen von Hardware-Ausfällen – einschließlich Stromausfällen – sicher und ermöglicht so ein schnelles Recovery mit möglichst geringen Auswirkungen auf die Benutzer.

Caché bietet zusätzlich fortgeschrittene Optionen für die Konfiguration der Hochverfügbarkeit, um Auswirkungen auf die Benutzer weiter zu reduzieren:

- Failover Cluster
- Shadow-Server
- Verteiltes ECP

### Failover Cluster

In einem Failover Cluster greifen mehrere Datenserver gemeinsam auf dieselben Platten zu, aber nur ein Server führt Caché aktiv aus. Wenn der aktive Server ausfällt, wird Caché automatisch auf einem anderen Server gestartet, der sofort die Verantwortung für die Verarbeitung übernimmt. Die Benutzer können sich ohne Zeitverzögerung direkt beim neuen Server anmelden.

### Shadow-Server

Der Caché Shadow-Server ist ein Backup-Server, der mit dem primären über TCP lose verbunden ist. Der primäre Server sendet fortlaufend sämtliche Datenbankaktualisierungen an den Shadow-Server, so dass dieser immer eine annähernd aktuelle Kopie der Datenbank besitzt. Das Umschalten auf den Shadow-Server erfolgt nicht so automatisiert wie bei einem Failover Cluster. Die Überlebensfähigkeit ist jedoch höher, weil die Hardware nicht physikalisch verbunden ist – der Shadow-Server kann sich sogar an einem anderen Ort befinden.

Shadow-Server und Failover Cluster lassen sich kombinieren, um die Fehlertoleranz weiter zu verbessern.

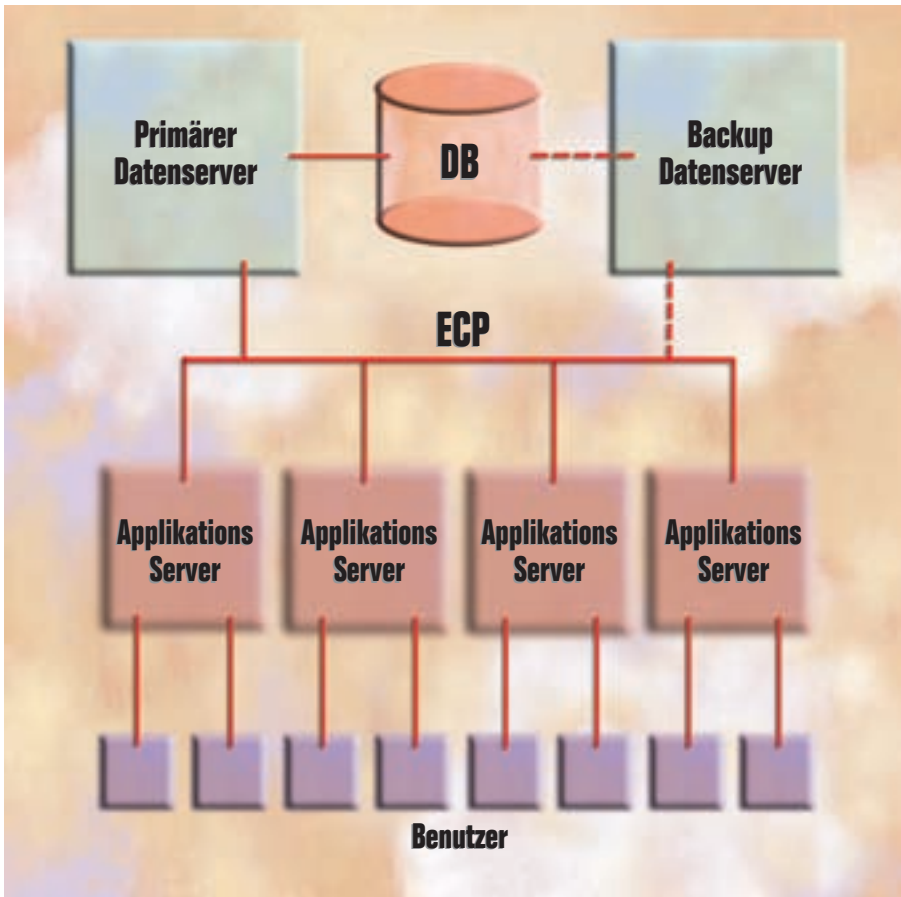
### Verteiltes ECP

Falls in verteilten Systemen, die mit ECP arbeiten, das Netzwerk kurz ausfällt oder der Datenserver abstürzt und neu bootet, versuchen die Applikationsserver sich neu anzumelden. Ist der Anmeldeversuch innerhalb einer festgesetzten Zeitspanne erfolgreich, sendet der Applikationsserver alle noch nicht bedienten Anforderungen erneut und der Betrieb wird fortgesetzt, wobei die Benutzer – von einer kleinen Pause abgesehen – nichts bemerken.

Wenn ein ECP-Applikationsserver ausfällt, sind nur die Benutzer des ausgefallenen Applikationsservers betroffen. Sie können sich dann bei einem anderen Applikationsserver anmelden und weiterarbeiten.

Ein ECP-Datenserver wird häufig als Failover Cluster konfiguriert. Wenn der primäre Datenserver abstürzt, übernimmt der Backup-Datenserver die Arbeit des ausgefallenen Servers und stellt somit einen unterbrechungsfreien Betrieb sicher, wobei die Benutzer wiederum nur eine kleine Pause bemerken.

**Ein ECP-Failover Cluster**



## **VORTEILE VON CACHE**

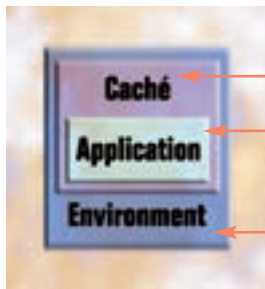
**Topsichere Datenbank:** Caché Write-Image Journaling und andere Funktionen gewährleisten die Datenbankintegrität für die meisten Arten von Hardware-Ausfällen einschließlich Stromausfällen.

**Hochverfügbare, fehlertolerante Konfigurationen:** Die Verwendung von Caché Shadow-Server, ECP und/oder Failover Cluster ermöglicht ein schnelles Recovery nach Ausfällen während die Auswirkungen für Benutzer minimiert werden und in einigen Fällen sogar ganz entfallen.



## SICHERHEITSMODELL

Caché verfügt über ein ausgefeiltes Sicherheitsmodell, das die Anwendungsentwicklung auf drei Arten unterstützt:



*Durch Sichern der Caché-Umgebung selbst.*

*Durch Sicherheitsfunktionen, die Entwickler einfach in ihre Anwendungen integrieren können.*

*Durch effektive Zusammenarbeit von Caché mit den Sicherheitstechnologien der Betriebsumgebung.*

Alle diese Sicherheitseinrichtungen von Caché beeinträchtigen die Performance von Anwendungen nur minimal.

### Benutzer, Rollen, Ressourcen und Privilegien

Caché unterscheidet verschiedene Ressourcen (wie Datenbanken, Anwendungen und System-Services), für die die Benutzer vom Security Administrator bestimmte Nutzungsberechtigungen erhalten (wie READ, WRITE oder USE). Zusätzlich zu den vom System definierten Ressourcen kann der Security Administrator anwendungsspezifische Ressourcen erstellen und für diese dieselben Mechanismen verwenden, um Berechtigungen zu gewähren und zu prüfen.

Der Einfachheit halber werden Benutzern eine oder mehrere „Rollen“ zugewiesen (z. B. „MTA“ oder „Lohnbuchhaltung“), und der Security Administrator vergibt dann Berechtigungen für eine bestimmte Ressource an die betreffende Rolle anstelle der einzelnen Benutzer. Der Benutzer erbt alle Privilegien, die der zugewiesenen Rolle gewährt werden.

Jeder Prozess verfügt über einen zugeordneten Benutzernamen, selbst wenn dies nur „UnknownUser“ ist. Der Benutzername wird während der „Authentifizierung“ zugewiesen. Das einfachste Beispiel für die Authentifizierung ist, wenn der Benutzer einen Benutzernamen und ein Passwort eingibt und das System prüft, ob das Passwort korrekt ist. Nach der Authentifizierung wird dem Prozess der Benutzername zugewiesen und die mit dem Benutzernamen verbundenen Berechtigungen gewährt. (Ein „Benutzer“ ist nicht notwendigerweise ein Mensch. Es kann beispielsweise auch ein Messinstrument sein, das Daten generiert, oder eine Anwendung, die auf einem anderen mit Caché verbundenem System läuft.) Wenn der Benutzer sich keiner Authentifizierung unterzieht, erhält er den Benutzernamen „UnknownUser“, der diesem Prozess nur die Berechtigungen erteilt, die unbekannt Benutzern gewährt werden.

Verbindungen zu Caché werden über verschiedene Services gesteuert. Jeder Service gibt an, ob er öffentlich ist – also jeder ihn verwenden kann – oder ob er eine Authentifizierung und bestimmte Zugriffsprivilegien erfordert. Services können einzeln deaktiviert werden, dann ist der Zugriff für jedermann gesperrt.

Die Zuweisung und das Management von Privilegien erfolgt normalerweise über das Caché Management Portal.

## Von Anwendungen zugewiesene Rollen

Es ist oftmals sinnvoll, einem Benutzer zusätzliche Privilegien temporär zur Verfügung zu stellen, statt sie dauerhaft zu erteilen. Anstatt einem Benutzer eine ganze Reihe von Privilegien zu erteilen wie die Erlaubnis, auf die Gehaltsliste zuzugreifen und diese zu bearbeiten, kann der Security Administrator dem Benutzer stattdessen nur das einzelne Privileg geben, auf die Anwendung der Gehaltsabrechnung zuzugreifen. Diese Anwendung kann dann während ihrer Nutzung dem Benutzer nach Bedarf weitere Privilegien zuweisen.

Hierfür können Anwendungen Rollen zugewiesen werden. Wenn dann der Benutzer mit dieser Anwendung arbeitet, erhält er seinerseits temporär die entsprechenden Rollen zugewiesen. Das kann eine einfache Liste mit Rollen sein, die dann für jeden Benutzer gilt, es sind aber auch komplexe Regeln möglich, die beispielsweise die einem Benutzer direkt zugewiesenen Rollen berücksichtigen.

Besonders nützlich ist diese Funktion für Browser-basierte Anwendungen, die mit CSP (Caché Server Pages) entwickelt wurden. Bei CSP gibt ein Teil jedes URL den Anwendungsnamen an. Nach der Authentifizierung und der Überprüfung, ob der Benutzer zur Verwendung der CSP-Anwendung berechtigt ist, erhält der Benutzer temporär weitere Rollen, die diese Anwendung für die Dauer der Seitenanforderung erteilt.

Der Security Administrator kann auch eigene Routinen angeben, die mittels selbst geschriebenem Code die Zuweisung der erweiterten Rollen durchführen, wenn die entsprechenden Sicherheitstests bestanden wurden. Nach diesem genau spezifizierten Mechanismus führen Nicht-CSP-Anwendungen die Rollenzuweisung durch.

## Authentifizierung

Caché unterstützt verschiedene Sicherheitsebenen zur Benutzerauthentifizierung, von keiner Authentifizierung über die Verwendung von Passwörtern bis hin zum Kerberos-Protokoll. Kerberos bietet eine sehr strenge Authentifizierung und hat den Vorteil, dass es schnell, skalierbar und einfach zu verwenden ist. Zudem werden bei Kerberos Passwörter nie über das Netzwerk übertragen, was ein zusätzliches Maß an Sicherheit darstellt.

Dabei unterstützt Caché auch „Single Sign-on“.



## Datenbankverschlüsselung

Caché bietet zwei Formen der Datenbankverschlüsselung:

- Der Security Administrator kann die Verschlüsselung für einzelne oder alle CACHE.DAT-Dateien (Datenbanken) aktivieren. Alles, was in diesen Dateien gespeichert ist, wird dann auf der Festplatte verschlüsselt.
- Entwickler können Systemfunktionen zum Verschlüsseln und Entschlüsseln von Daten verwenden, die in der Datenbank gespeichert oder übertragen werden sollen. So können sensitive Daten verschlüsselt werden, um diese vor anderen Benutzern zu schützen, die Lesezugriff auf die Datenbank haben, aber nicht über den entsprechenden Schlüssel verfügen.

Standardmäßig verschlüsselt Caché Daten mit einer Implementierung des Advanced Encryption Standard (AES), eines symmetrischen Algorithmus, der Schlüssel mit 128, 192 oder 256 Bit unterstützt. Die Verschlüsselungsschlüssel werden an einem geschützten Speicherort aufbewahrt. Caché enthält die gesamte für die Schlüsselverwaltung erforderliche Funktionalität.

Auch das Journal kann verschlüsselt werden.

## Auditing

Viele Anwendungen, vor allem solche, die gesetzliche Bestimmungen wie HIPAA oder Sarbanes-Oxley erfüllen müssen, erfordern ein sicheres Auditing. In Caché werden alle System- und Anwendungs-events in einem vor Veränderungen geschützten Log aufgezeichnet. Dieser Log ist zu allen mit SQL arbeitenden Abfragewerkzeugen oder Listengeneratoren kompatibel.

## Kapitel 3: Der Caché-Applikationsserver

Der Applikationsserver von Caché bietet alle Möglichkeiten modernster Objektprogrammierung, verfügt über intelligente Caching-Algorithmen und schafft den einfachen Zugang zu einer Vielfalt von Technologien. Der Applikationsserver ermöglicht die schnelle Entwicklung anspruchsvoller Datenbankanwendungen, die zur Laufzeit höchste Performance und einfache Administration bieten.

Der Applikationsserver von Caché zeichnet sich durch folgende Leistungsmerkmale aus:

- Zugriff auf die multidimensionalen Datenserver von Caché auf dem gleichen und anderen Rechnern mit transparenter Weiterleitung der Daten.
- Konnektivität mit Caching auf der Client-Seite, um den schnellen Zugriff auf Caché-Objekte aus allen gängigen Technologien wie Java, C++, COM, .NET und Delphi bereitzustellen. Caché sorgt dabei automatisch für die Vernetzung zwischen Client und Applikationsserver.
- Kompatibilität zu SOAP und XML.
- SQL-Zugriff mit ODBC und JDBC. Ein intelligentes Caching auf Client und Applikationsserver sichert höchste Performance.
- Zugriff auf relationale Datenbanken.
- Caché Server Pages garantieren hochperformante und einfach zu erstellende Web-Anwendungen.
- Caché Studio – eine IDE zur schnellen Entwicklung und dem Debuggen von Anwendungen mit Caché.
- Code der Skriptsprachen wird in der Datenbank gespeichert und kann online geändert werden, wobei die Änderungen automatisch an alle Applikationsserver weitergeleitet werden.

### DIE CACHÉ VIRTUAL MACHINE UND SKRIPTSPRACHEN

Im Zentrum des Caché-Applikationsservers steht die extrem schnelle Caché Virtual Machine, die die Skriptsprachen von Caché unterstützt:

- Caché ObjectScript ist eine leistungsstarke und einfach zu erlernende objektorientierte Sprache mit extrem flexiblen Datenstrukturen.
- Caché Basic erleichtert Visual-Basic-Programmierern den Einstieg in Caché. Es ist VBScript ähnlich und verfügt über Erweiterungen, die den direkten Zugriff auf die multidimensionalen Arrays von Caché ermöglichen.
- Caché MVBasic ist die Variante der Programmiersprache Basic, die in MultiValue (Pick)-Anwendungen verwendet wird. Das MVBasic von Caché wurde um Objektunterstützung erweitert und ermöglicht den direkten Zugriff auf die multidimensionalen Arrays von Caché.

In der Caché Virtual Machine ist der Datenbankzugriff hochoptimiert. Jeder Benutzerprozess kann hier direkt auf die multidimensionalen Datenstrukturen zugreifen, indem Aufrufe an den gemeinsamen Speicher abgesetzt werden, der wiederum auf einen gemeinsamen Datenbank-Cache zugreift. Alle anderen Technologien (Java, C++, ODBC, JDBC usw.) melden sich über die Caché Virtual Machine für den Zugriff auf die Datenbank an.

## Umfassende Interoperabilität

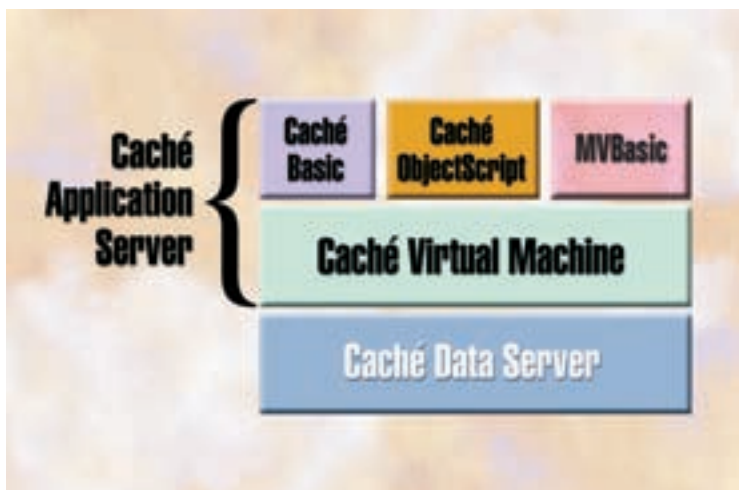
Da Caché ObjectScript, Basic und MVBasic auf derselben Caché Virtual Machine implementiert sind, sind sie vollständig interoperabel:

- Jede Objektmethode kann in einer beliebigen Sprache erstellt werden – in ein und derselben Klasse lassen sich alle drei Sprachen kombiniert einsetzen.
- Jede Sprache kann Code aufrufen, der in den anderen Sprachen erstellt wurde.
- Variablen, Arrays und Objekte werden gemeinsam genutzt.

## Schnellere Entwicklung / flexible Bereitstellung

Je mehr Code Programmierer in den Skriptsprachen erstellen und somit auf eine Ausführung der Anwendungen in der Caché Virtual Machine setzen, um so schneller wird in der Regel die Entwicklung von Anwendungen. Darüber hinaus laufen die Anwendungen schneller und sind höher skalierbar. Und schließlich ist bei einem Wechsel der Hardwarekonfiguration oder des Betriebssystems keine Änderung am Code erforderlich – Caché sorgt automatisch für die Berücksichtigung aller Unterschiede in Betriebssystem und Hardware.

### Skriptsprachen



## VORTEILE VON CACHÉ

### Schnelle Anwendungsentwicklung:

Mit Caché ObjectScript lassen sich komplexe Datenbankanwendungen bedeutend schneller entwickeln als in anderen verbreiteten Programmiersprachen – häufig 10- bis 100-mal schneller. Schnellere Entwicklung bedeutet auch, dass das Projekt höhere Erfolgschancen hat – mit weniger Programmierern – und sofort angepasst werden kann, wenn sich die Anforderungen ändern.

**Kürzere Lernkurve:** Basic ist wahrscheinlich die weltweit bekannteste Computersprache. Entwickler, die Visual Basic kennen, können sofort Code in Basic schreiben, und das Objektmodell von Caché lässt sich schnell erlernen.

### Schneller und höher skalierbar:

Die Caché Virtual Machine mit ihrem direkten Zugriff auf die Datenbank führt zu schnelleren Anwendungen, die bis zu Zehntausende von Benutzern auf preiswerter Hardware skalieren können.

**Flexibilität:** Code, der auf der Caché Virtual Machine läuft, kann unverändert auch auf anderer Hardware und anderen Betriebssystemen ausgeführt werden. Code wird in der Datenbank gespeichert und automatisch an die Applikationsserver weitergegeben.

## CACHÉ OBJECTSCRIPT

Caché ObjectScript ist eine leistungsstarke objektorientierte Programmiersprache, zugeschnitten auf die schnelle Entwicklung von Datenbankanwendungen. Im Folgenden werden grundlegende Eigenschaften der Sprache erläutert.

### Grundstruktur

Caché ObjectScript ist befehlsorientiert, wie folgende Syntax-Beispiele zeigen:

```
set x=a+b
do rotate(a,3)
if (x>3)
```

Es gibt eine Anzahl integrierter Systemfunktionen, die sich besonders für die Textbearbeitung eignen. Ihre Namen beginnen alle mit einem '\$'-Zeichen, um sie von den Variablen- und Array-Namen zu unterscheiden. Zum Beispiel:

```
$extract(string,von,bis) // liefert bestimmte Zeichen aus einer Zeichenkette
$length(string) // bestimmt die Länge der Zeichenkette
```

Analog der Arbeitsweise der meisten Taschenrechner werden Ausdrücke streng von links nach rechts abgearbeitet, es sei denn, die Reihenfolge wird mit Klammern geändert.

### Flexible Datenspeicherung

Caché ObjectScript zeichnet sich besonders durch die sehr flexible und dynamische Datenspeicherung aus. Daten können gespeichert werden als:

- Objekteigenschaften
- Variablen
- multidimensionale Sparse-Arrays mit Subscripts beliebiger Datentypen
- Datenbankelemente („Globals“), d. h. persistente multidimensionale Sparse-Arrays

Mit ganz wenigen Ausnahmen kann an jeder Stelle der Sprache anstelle einer Variablen auch ein Array, eine Objekteigenschaft oder ein Global verwendet werden.

In den meisten Programmiersprachen sind Datentypen eine Erweiterung der Hardware-bezogenen Speicherkonzepte (Integer, Float, Char etc.). Caché ObjectScript geht dagegen von der Philosophie aus, dass Menschen nicht in solchen Kategorien zur Informationsspeicherung denken und deshalb solche Computer-zentrierten Datentypen einer produktiven Anwendungsentwicklung ganz einfach im Weg stehen. Der Gebrauch von Deklarationen und Dimensionsanweisungen verursacht mehr Fehler, als er zu vermeiden hilft (man denke nur an den Überlauf eines 2-Byte-Integers oder an das mögliche Problem, dass eine Zeichenkette den zugewiesenen Speicher überschreitet und anderen Speicher damit beschädigt). Eine Objekttypisierung, wie z. B. „Person“, „Rechnung“, „Tier“, „Fahrzeug“ etc. ist dagegen dem menschlichen Denken angemessen und intuitiv handhabbar.

Deshalb werden in Caché ObjectScript Objekteigenschaften stark typisiert, die anderen drei Typen der Datenspeicherung (Variablen, Arrays und Globals) sind dagegen polymorphe, typenlose Entitäten, die keiner Deklaration oder Definition bedürfen. Sie treten ganz einfach dann in Erscheinung, wenn sie gebraucht werden, und nehmen genau die Form an, die für die Speicherung der jeweiligen Daten und die Verwendung in einem Ausdruck benötigt wird. Nicht einmal Arrays bedürfen einer Spezifikation hinsichtlich ihrer Größe und Dimension oder der Typen von Subscripts und Daten. So könnte ein Entwickler ein Array „Person“ einfach folgendermaßen anlegen:



```
set Person("Schmid","Hans")="Ich bin ein guter Mensch."
```

In diesem Beispiel werden Daten in einem zweidimensionalen Array mit Strings als Subscripts gespeichert. Andere Datenknoten dieses Arrays könnten über mehr Dimensionen verfügen oder Strings, Integer-Werte und weitere Datentypen in den Subscripts mischen. Zum Beispiel könnten Daten so gespeichert werden:

```
abc(3)  
abc(3,-45.6,"Ja")  
abc("Zähler")
```

Alle Daten werden im selben Array gespeichert.

### Direkter Datenbankzugriff

Eine direkte Referenz auf die Datenbank (eine "Global-Referenz") ist im Grunde eine multidimensionale Array-Referenz, vor der das ^-Zeichen steht. Dieses Zeichen gibt an, dass es sich um eine Referenz auf die in der Datenbank gespeicherten Daten handelt und nicht um die temporäre Verarbeitung lokaler Daten. Ein solches Datenbank-Array wird dann als "Global" bezeichnet.

Wie bei multidimensionalen Arrays und Variablen sind keine Deklarationen, Definitionen oder Speicherreservierungen erforderlich, um auf die Daten in der Datenbank zuzugreifen oder sie zu speichern. Global-Daten treten einfach in Erscheinung, wenn die Daten erstmals gespeichert werden. Um Informationen in der Datenbank zu speichern, könnte man beispielsweise Folgendes schreiben:

```
set ^Person("Schmid","Hans")="Ich bin ein sehr guter Mensch."
```

und später es mit Code wie dem folgenden wieder abrufen:

```
set x=^Person("Schmid","Hans")
```

Der Programmierer ist bei der Strukturierung dieser Global-Daten-Arrays vollkommen frei. (Siehe auch „Das multidimensionale Datenmodell“)

## Objektreferenzen

Caché Objects implementieren das ODMG-Datenmodell mit einigen mächtigen Erweiterungen.

In Caché ObjectScript wird mit einer Objektreferenz auf ein Objekt zugegriffen (eine Objektreferenz ist in der Regel eine Variable, deren Wert angibt, auf welches Objekt im Speicher verwiesen wird).

Auf die Objektreferenz folgt ein Punkt und dann der Name der Eigenschaft oder der Methode.

Objektreferenzen können immer anstelle von Ausdrücken verwendet werden. Ein Beispiel:

```

set name=person.Name      // 'person' ist eine Variable, deren Wert eine
                           // Objektreferenz ist.
                           // Der Name der Person wird in die Variable 'name'
                           // geschrieben.

if (person.Alter>x)       // Prüfe, ob das Alter der Person über 'x' liegt.
set betrag=rechnung.Summe() // 'Summe()' ist eine Methode, die die Summe
                           // aller Rechnungspositionen ermittelt.

```

Methoden können auch mit einem DO-Befehl ausgeführt werden, wenn kein Rückgabewert benötigt wird. Zum Beispiel speichert:

```

do artikel.Erhöhe()       // 'Erhöhe()' ist eine Methode, deren Rückgabewert
                           // falls vorhanden, nicht relevant ist.

```

Man darf die Objektreferenz nicht mit der Objekt-ID verwechseln. Die Objekt-ID ist ein Wert, der einem Datenbankobjekt fest zugeordnet ist. Sie dient zum Abruf und zur Speicherung eines Datenbankobjekts. Befindet sich ein Objekt im Speicher, wird ihm ein wiederverwendbarer Objektreferenzwert zugewiesen, mit dem man dann auf die Objektdaten zugreift. Wenn dasselbe Datenbankobjekt das nächste Mal in den Speicher geladen wird, wird ihm höchstwahrscheinlich eine andere Objektreferenz zugewiesen.



## HTML- und SQL-Zugriff

In Caché ObjectScript-Code kann HTML für Web-Anwendungen sowie SQL eingebettet werden.

### Code aufrufen

In einigen objektorientierten Sprachen muss der gesamte Code immer Teil einer Methode sein. Caché ObjectScript verzichtet auf diese Einschränkung – Code kann direkt oder über die Objektsyntax aufgerufen werden. Code wird häufig mit dem DO-Befehl aufgerufen.

```
do rotate(a,3)
```

Code, der einen Wert liefert, kann auch als Funktion aufgerufen werden. Ein Beispiel:

```
set x=a+$$insert(3,y)
```

ruft die vom Programmierer erstellte Prozedur oder Subroutine “insert” auf.

Code kann auch als Objektmethode aufgerufen werden.

```
set betrag=rechnung.Summe() // Summe() liefert den Gesamtbetrag der Rechnung  
artikel.Erhöhe()           // 'Erhöhe()' ist eine Methode, deren Rückgabewert  
                           // falls vorhanden, nicht relevant ist.
```

Für Parameter wird sowohl die Übergabe von Werten (Call by Value) wie auch die Übergabe von Referenzen (Call by Reference) unterstützt.



## Routinen

Mit Caché ObjectScript erstellter Code wird prinzipiell in "Routinen" organisiert. Jede Routine, die in der Regel bis zu 32 KB groß werden kann, bildet eine elementare Einheit und kann unabhängig von anderen Routinen editiert, gespeichert, kompiliert und ausgeführt werden. Die Verknüpfung von Routinen erfolgt dynamisch zur Laufzeit; Programmierer müssen sich nicht zusätzlich darum kümmern. Der Code von Routinen ist in der Datenbank gespeichert; so können sie dynamisch im gesamten Netzwerk aufgerufen werden und müssen nicht auf jedem einzelnen Rechner installiert werden.

Der Code einer Routine wiederum ist als ein Satz von Prozeduren und/oder Subroutinen aufgebaut. (Eine Objektmethode ist ebenfalls eine Prozedur, auf sie wird aber über eine spezielle Syntax zugegriffen.)

Beim Aufruf von Code, der sich in der gleichen Routine befindet, wird nur der Name der Prozedur oder der Subroutine benötigt. Anderenfalls muss der Routinenname angehängt werden.

```
do transfer()           // ruft "transfer" in derselben Routine auf.
do summe^rechnung()    // ruft "summe" in der Routine "rechnung" auf.
```

Ist der zurückgegebene Wert einer Prozedur oder einer Subroutine relevant, sollte der Aufruf über die Funktionssyntax „\$\$“ erfolgen.

```
set x=$$summe^rechnung() // Ruft die gleiche Prozedur "summe" auf, verwendet
                          // aber den Rückgabewert
```

Routinen können in Caché Studio bearbeitet und kompiliert werden.



## Objektmethoden

Klassendefinitionen und deren Methoden-Code werden in Globals gespeichert; der Class Compiler kompiliert dann jede Klasse in eine oder mehrere Routinen. Jede Methode ist einfach eine Prozedur in einer Routine, obwohl sie nur über die Objektsyntax aufgerufen werden kann. Wenn die Klasse „Patient“ beispielsweise eine Methode „Aufnahme“ definiert und die Variable „Pat“ eine Referenz auf ein konkretes Objekt „Patient“ darstellt, dann rufen wir die Methode „Aufnahme“ für dieses Objekt mit der folgenden Syntax auf:

```
do Pat.Aufnahme()           // Aufruf der Methode Aufnahme für Patient
set x = Pat.Aufnahme()      // Ruft die gleiche Methode auf, verwendet aber den
                             Rückgabewert.
```

## Prozeduren und öffentliche bzw. private Variablen

Eine Prozedur ist ein Code-Block in einer Routine und entspricht in etwa einer Funktion in anderen Programmiersprachen. Prozeduren bestehen aus einem Namen, einer Liste von formalen Parametern, einer Liste von öffentlichen Variablen und einem Code-Block, der durch ‘{ }’ begrenzt wird.

Zum Beispiel:

```
Aufnahme(x,y)[name,pid]    { ...hier folgt der Code.
                             }
}
```

In Caché ObjectScript sind Variablen entweder öffentlich (allgemein verfügbar) oder nur innerhalb einer bestimmten Prozedur sichtbar, also privat. Jede in einer Prozedur verwendete Variable wird als privat für diese Prozedur betrachtet, sofern sie nicht in der öffentlichen Liste aufgeführt wird. Im obigen Beispiel sind „name“ und „pid“ öffentliche Variablen, wohingegen alle anderen Variablen nur während der Ausführung dieser Prozedur existieren. Variablennamen, die mit dem Prozentzeichen ‘%’ beginnen, sind implizit öffentlich.

Prozeduren können nicht verschachtelt werden, aber eine Prozedur kann Subroutinen enthalten.

## Subroutinen

Routinen können auch Subroutinen enthalten, die weniger strikt als Prozeduren sind. Eine Subroutine kann eine Parameterliste enthalten und einen Wert zurückgeben. Sie hat aber keine Public-Liste und keine formale Blockstruktur. Subroutinen können in Prozeduren eingebettet werden oder sich auf der gleichen Ebene wie eine Prozedur in einer Routine befinden.

Subroutinen erlauben den Aufruf von Code mit der gleichen öffentlichen/privaten Variablenmenge wie das aufrufende Programm, und sie können schneller aufgerufen werden als Prozeduren. Eine Subroutine, die in einer Prozedur eingebettet ist, verwendet denselben Variablenbereich wie die Prozedur und kann nur in der Prozedur aufgerufen werden. Variablenreferenzen einer Subroutine, die nicht zu einer Prozedur gehört, verweisen alle auf öffentliche Variablen.

## BASIC

Basic ist wahrscheinlich die weltweit bekannteste Sprache zur Anwendungsentwicklung. In Caché wurde Basic um den direkten Zugriff auf die zentralen Datenstrukturen – die multidimensionalen Arrays – des Datenservers wie auch um andere Funktionen des Applikationsservers von Caché erweitert. Es unterstützt direkt das Objektmodell von Caché über die Syntax von Visual Basic und läuft in der Caché Virtual Machine.

Basic kann entweder in Form von Klassenmethoden oder als Caché-Routinen zum Einsatz kommen (vgl. die Beschreibung der Routinen von Caché ObjectScript). Basic kann Caché ObjectScript aufrufen und umgekehrt, wobei beide Sprachen auf die gleichen Variablen, Arrays und Objekte im Prozessspeicher zugreifen.

Die Funktionsweise von Arrays wurde erheblich erweitert:

- Steht vor dem Array-Namen das ^-Zeichen, so weist dies auf eine Referenz auf ein multidimensionales Datenbank-Array hin – solche „Globals“ werden mit anderen Prozessen gemeinsam genutzt.
- Die Datentypen von Subscripts können frei gewählt werden – Strings, Integer, Dezimalzahlen usw.
- Daten können auf mehreren Subscript-Ebenen im selben Array gespeichert werden – beispielsweise unter A(„Farben“) und A(„Farben“,3).
- Arrays müssen nicht deklariert werden und sind immer sogenannte Sparse Arrays, für die Caché nur Platz reserviert, wenn Knoten eingefügt werden.
- Die Funktion „Traverse“ erlaubt die Ermittlung des nächsten (oder vorherigen) Subscripts auf einer bestimmten Subscript-Ebene.

Zu den Erweiterungen gehören darüber hinaus:

- Transaktionsverarbeitungsbefehle, um eine Transaktion zu starten (Start), festzuschreiben (Commit) und zurückzurollen (Rollback).
- Eine atomare Inkrementierungsfunktion, die auf die Datenbank angewendet werden kann.
- Erweiterungen, die eine bessere Integration in die Applikationsserver-Funktionalitäten von Caché ermöglichen.



## Objektzugriff mit Basic

In Caché werden Klassen in Packages strukturiert und die Klassennamen beinhalten den Package-Namen gefolgt von einem Punkt. So ist beispielsweise „Lohnbuchhaltung.Person“ die Klasse „Person“ des Packages „Lohnbuchhaltung“. Mit dem Basic-Befehl „New“ wird ein Objekt angelegt:

```
person = New Lohnbuchhaltung.Person() // Erstellt ein neues Objekt „Person“.
```

Um auf ein bestehendes Objekt zugreifen zu können, wurde Basic um einen OpenID-Befehl erweitert:

```
person = OpenID Lohnbuchhaltung.Person(54) // Öffnet das Objekt „Person“ mit der  
Objekt-ID 54.
```

Hier einige Code-Beispiele, die auf die Eigenschaften von „Person“ zugreifen:

```
person.Name = "Schmid, Hans" // Legt den Namen der Person fest.  
person.Privatanschrift.Ort // Gibt den Wohnort der Person an.  
person.Arbeitgeber.Name // Holt das Objekt „Arbeitgeber“ der  
Person  
// und greift auf den Namen des  
Arbeitgebers zu.
```

Datenbankklassen können mit der Methode „Save“ auf Festplatte gespeichert werden:

```
person.Save()
```

speichert beispielsweise eine Person. Falls das Objekt das erste Mal gespeichert wird, wird eine Objekt-ID vergeben. Geänderte verknüpfte Objekte (wie „Arbeitgeber“) werden ebenfalls automatisch gespeichert.



## MVBASIC

MVBasic, eine Basic-Variante, ist eine weitere Skriptsprache von Caché. Sie ist zur Ausführung von Anwendungen gedacht, die für MultiValue (Pick)-Systeme erstellt wurden, und unterstützt daher zusätzliche Funktionalität zum Zugriff auf und zur Bearbeitung von MultiValue-Dateien.

MVBasic kann entweder in Form von Klassenmethoden oder als Caché-Routinen zum Einsatz kommen (vgl. die Beschreibung der Routinen von Caché ObjectScript). MVBasic kann Caché ObjectScript oder Basic aufrufen und umgekehrt, wobei alle drei Sprachen auf die gleichen Variablen, Arrays und Objekte im Prozessspeicher zugreifen.

Caché MVBasic weist dieselben Erweiterungen wie Caché Basic einschließlich des Objektzugriffs auf. Auf Grund der möglichen Mehrdeutigkeit wird aber in Objektreferenzen die Zeichenfolge „->“ anstelle des Punkttrennzeichens „.“ verwendet.

## C++

Jede Caché-Klasse kann als C++-Klasse projiziert werden, wobei die Methoden den einzelnen Eigenschaften und Methoden der Caché-Klasse entsprechen. Für C++-Programme sehen diese Klassen wie jede andere lokale C++-Klasse aus. Caché sorgt automatisch für die gesamte Kommunikation zwischen Client und Server. Die Objekteigenschaften werden auf dem Client im Cache zwischengespeichert. C++-Methodenaufrufe rufen die entsprechenden Methoden auf der Server-Seite auf – einschließlich Methoden, um ein Objekt in der Datenbank zu speichern und es später wieder aufzurufen.

## JAVA

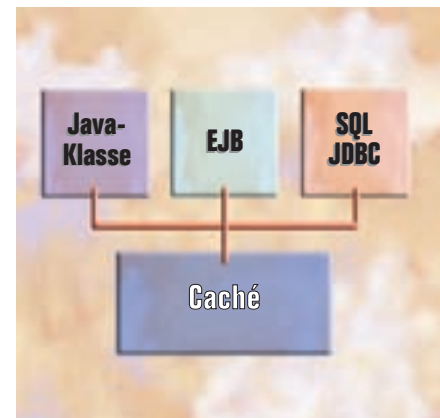
Die Programmiersprache Java ist sehr populär, aber die Anbindung von Java-Anwendungen an die meisten Datenbanken stellt eine Herausforderung dar. Die Anbindung an eine relationale Datenbank erfordert üblicherweise umfangreiche SQL-Programmierung. Dies ist sehr zeitintensiv und relativiert viele Vorteile der Objekttechnologie von Java. Der Ansatz von Caché, Objekte direkt zu speichern, ohne dass sich die Entwickler darum kümmern müssen, wie die Daten persistent abgelegt werden, und die Verwendung der Objektsyntax für den Zugriff auf die Datenbank sind viel einfacher – und in der Regel die bevorzugte Lösung.

Einige Entwickler arbeiten ausschließlich mit gewöhnlichen Java-Objekten („plain old Java objects“, POJO), während andere Enterprise Java Beans (EJB) bevorzugen. Manche Entwickler definieren lieber zuerst das Datenbankschema und generieren dann automatisch die entsprechende Java-Klasse für jede Datenbankklasse, während andere lieber zuerst die Java-Klassen erstellen und dann mit Caché automatisch ein Datenbankschema generieren.

Caché unterstützt alle diese Ansätze:

- Jede Caché-Klasse kann als Java-Klasse projiziert werden, so dass auf die Eigenschaften und Methoden als Java-Objekte zugegriffen werden kann.
- Caché-Klassen können auch als Enterprise Java Beans (EJB) projiziert werden.
- JDBC bietet einen High-Performance SQL-Zugriff durch einen vollständig Java-basierten (Typ 4) Treiber.
- Die Jalapeño-Technologie von InterSystems erstellt Caché-Klassen aus vorhandenen POJO-Klassenbeschreibungen.

### Unterschiedliche Arten der Java-Unterstützung



## Objektzugriff über projizierte Klassen

Jede Caché-Klasse kann als eine Java- (oder EJB-) Klasse projiziert werden, wobei die Methoden den einzelnen Eigenschaften und Methoden der Caché-Klasse entsprechen. Für Java-Programme sehen diese Klassen genauso aus wie jede andere lokale Java-Klasse. Die erzeugte Java-Klasse verwendet eine von InterSystems bereitgestellte Java-Bibliothek, um die gesamte Kommunikation zwischen Client und Server abzuwickeln.

Der Zustand jedes Caché-Objekts wird im Applikationsserver verwaltet, obwohl die Eigenschaften der Klasse auch auf dem Client im Cache zwischengespeichert werden, um die Performance zu optimieren. Java-Methodenaufrufe rufen die entsprechenden Methoden auf dem Applikationsserver von Caché auf – einschließlich Methoden zur Speicherung eines Objekts in der Datenbank und zum späteren Wiederaufrufen. Für den Client ist es transparent, welcher Datenserver die Daten enthält, selbst dann, wenn die Objektdaten in einer relationalen Datenbank gespeichert werden, auf die über den Applikationsserver von Caché zugegriffen wird.

## In Java geschriebene Caché-Methoden

Methoden der Caché-Klassen können im Caché Studio in Java geschrieben werden. Im Unterschied zu Caché ObjectScript und Basic werden Java-Methoden nicht von der Caché Virtual Machine ausgeführt. Stattdessen werden sie in die erzeugte Java-Klasse mit aufgenommen und können auf jeder beliebigen Java Virtual Machine ausgeführt werden. Auf solchen Code kann nur über Java zugegriffen werden.

## Persistenz für J2EE-Anwendungen

Entwickler von J2EE-Anwendungen, die Enterprise Java Beans (EJB) einsetzen, arbeiten in erster Linie mit Objekten. Wenn sie allerdings auf die Datenbank zugreifen wollen, müssen sie normalerweise auf SQL zurückgreifen. Über seine JDBC-Schnittstelle kann Caché solchen Anwendungen einen extrem schnellen SQL-Zugriff bieten. Trotzdem ist der Zugriff über SQL nicht der allgemein empfohlene Ansatz.

Die natürlichere Zugriffstechnik für EJB-Programmierer stellen Objektdatenbanken dar. Caché projiziert dazu Caché-Klassen als EJBs, wobei automatisch leistungsstarke Persistenzmethoden für die Bean-Managed Persistence (BMP) generiert werden. Hierdurch werden der Overhead für SQL und das objektrelationale Mapping komplett vermieden – das Ergebnis: eine wesentlich verbesserte Skalierbarkeit für J2EE-Anwendungen.

Die Jalapeño-Technologie von InterSystems kann mit denselben Vorteilen auch in J2EE-Anwendungen eingesetzt werden.

## VORTEILE VON CACHE

**Flexibilität:** Beim Zugriff auf Caché-Objekte haben Java-Entwickler die freie Wahl – sie können SQL und JDBC einsetzen oder natürlich auch Objekte als Java-Klassen oder Enterprise Java Beans projizieren. Mit Jalapeño können Entwickler vollständig in ihrer bevorzugten Java-Entwicklungs-umgebung arbeiten. Caché stellt dann automatisch Methoden zur Verfügung, um die Objekte aus der Datenbank abzurufen und zu speichern, ohne die Klassen der Entwickler anzurühren.

**Höchste Performance:** Alle Java-Anwendungen profitieren von der überlegenen Performance und Skalierbarkeit von Caché, unabhängig davon, wie sie sich mit Caché verbinden.

**Native Kompatibilität zu J2EE für beschleunigte Entwicklung:** Caché-Klassen können sehr einfach als EJBs projiziert werden, wodurch J2EE-Entwickler ganz natürlich mit der postrelationalen Datenbank Caché arbeiten können. Wird eine Caché-Klasse unter Verwendung der Bean-managed Persistence projiziert, generiert Caché automatisch die Methoden zum Zugriff auf die Caché-Datenbank. Da die Entwickler Persistenzmethoden nicht mehr länger von Hand schreiben müssen, können Anwendungen schneller fertig gestellt werden.

## Jalapeño ermöglicht Entwicklung von Java aus

Anstatt mit Caché-Klassen zu beginnen und diese als Java-Komponenten zu projizieren, funktioniert die Jalapeño-Technologie von InterSystems genau andersherum. Mit ihr können Java-Entwickler Objektklassen in einer beliebigen Java-Entwicklungsumgebung definieren und diese Klassen dann automatisch in Caché persistent machen. Die Java-Klasse des Entwicklers wird nicht verändert – Caché bietet eine Bibliotheksklasse mit einem API, die für das Laden und Speichern der Objekte und das Ausführen von Abfragen für die Entwicklerklassen sorgt.

## CACHÉ UND JALAPEÑO

Java-Entwickler, die neue Datenbankanwendungen erstellen möchten, stehen heutzutage vor verschiedenen Problemen. In der Regel speichern sie ihre Daten mit SQL in einer relationalen Standarddatenbank. Bei diesem Ansatz müssen Entwickler ihre Java-Objekte auf die relationalen Strukturen abbilden und langwierige und oftmals komplexe SQL-Abfragen erstellen, die auf diese Daten zugreifen – was leicht einen Großteil der Gesamtentwicklungszeit beanspruchen kann. Alternativ kann ein Entwickler eine objektorientierte Datenbank verwenden, die oftmals weder SQL noch SQL-basierte Abfragewerkzeuge unterstützt und meist umfangreiche Schemadefinitionen erfordert.

Jalapeño (**J**AVA **L**anguage **P**ersistence with **NO** mapping) ist eine InterSystems-Technologie, mit der Java-Entwickler einfach ihre Objekte in der robusten Caché-Datenbank mit objektorientiertem Zugriff persistent machen können, während gleichzeitig hochperformanter SQL-Zugriff auf dieselben Daten möglich ist. Objekte werden in der Datenbank als echte Objekte mit Eigenschaften, Beziehungen usw. gespeichert (also nicht einfach durch Speicherung des serialisierten Zustands), dennoch ist kein objekt-relationales Mapping erforderlich.

Mit Jalapeño erstellen Java-Entwickler Datenbankklassen in ihrer bevorzugten Java-Entwicklungsumgebung auf dieselbe Weise wie alle anderen POJO-Klassen. Der Entwickler definiert dann für Jalapeño, welche Klassen Datenbankklassen sind – in der Regel mit Hilfe eines Plug-ins für die Entwicklungsumgebung, das in der Jalapeño Persistence Library enthalten ist. Jalapeño analysiert die Klassen, erstellt automatisch das entsprechende Objekt- (und SQL-)Datenbankschema und erzeugt die gesamte Laufzeitunterstützung zum Speichern und Abrufen solcher Objekte.

Die POJO-Klasse des Entwicklers wird nicht verändert. Der Entwickler kann sie weiterhin modifizieren.

Zur Laufzeit greift die Anwendung wie gewohnt direkt auf die Eigenschaften und Methoden des POJO-Objekts zu. Um Datenbankobjekte zu speichern und abzurufen, verwendet die Anwendung die APIs der von Jalapeño bereitgestellten ObjectManager-Klasse. Diese Klasse stellt auch Methoden bereit, die eine Verbindung zur Datenbank erstellen, SQL-Abfragen unterstützen und Transaktionssemantik wie „Start“, „Commit“ und „Rollback“ bieten.

Eine einfache Klassenbeschreibung mit einer Liste der Eigenschaften und deren jeweiligem Typ allein ist nicht ausreichend, um eine Datenbankspeicherung daraus abzuleiten. Zumindest muss noch angegeben werden, welche Eigenschaft die Objekt-ID enthält. Entwickler wollen in der Regel auch Indizes angeben, um Abfragen effizienter zu gestalten. Solche und weitere Informationen für die Datenbankgenerierung können als Java-Annotationen in den Quelldateien angegeben werden.

Jalapeño unterstützt die “Schemaweiterentwicklung”, durch die der Entwickler die Klassen weiter verändern kann. Es können neue Eigenschaften hinzugefügt oder Eigenschaftsdefinitionen geändert werden. Die Schemadefinition passt sich dabei entsprechend an, ohne dass bereits eingegebene Daten ungültig werden. Dies führt zu einem natürlichen, iterativen Entwicklungsprozess.

Wenngleich Jalapeño am besten mit Caché funktioniert, kann es mit Hilfe der Standard-DDL (Data Definition Language) das erzeugte Datenbankschema auch in ein entsprechendes relationales Schema exportieren. So kann eine Anwendung, die für den Objektzugriff mit Caché erstellt wurde, auch auf einer relationalen Datenbank betrieben werden. In diesem Fall verwenden die ObjectManager-APIs von Jalapeño automatisch JDBC-Standardaufrufe für die Verbindung zur Datenbank. Zur Verbindung mit der Caché-Datenbank wird jedoch ein leistungsstärkeres, objektbasiertes Protokoll verwendet.

Die Jalapeño-Bibliothek selbst ist in Standard-Java implementiert und kann auf einer beliebigen Java 1.5 (oder höher) JVM oder J2EE-Applikationsserver-Umgebung ablaufen.

Bei Jalapeño kann sich der Java-Entwickler ganz auf die Benutzeroberfläche und die Geschäftslogik der Anwendung konzentrieren und Datenbankklassen einfach auf dieselbe Weise wie andere Klassen erstellen – Caché erledigt den Rest.

Im folgenden Beispiel wird ein Objekt „Kunde“ abgerufen, die Telefonnummer mit einer “set”-Methode geändert, die Datenbank aktualisiert und das Objekt im Speicher wieder geschlossen:

```
Kunde kunde = (Kunde) objectManager.openById(Kunde.class, kundeId);
kunde.setTelefonNummer("06151/1747-0");
objectManager.update(kunde, true);
objectManager.close();
```

## VORTEILE VON CACHE

**Schnellere und natürlichere Entwicklung ohne objekt-relacionales Mapping:** Jalapeño untersucht die Java-Klassen, um automatisch ein objektorientiertes Datenbankschema zu erzeugen. Die Daten werden als Objekte gespeichert und zusätzlich auch eine SQL-Standarddarstellung erstellt. Es ist kein objekt-relacionales Mapping erforderlich, was die Entwicklung beschleunigt.

**Einfache POJO-Persistenz:** In einer Anwendung greifen Entwickler auf ihre Datenbankklassen wie auf alle anderen Klassen zu. Jalapeño generiert den gesamten Code, um Objekte in der Datenbank zu speichern und daraus abzurufen. Hierfür werden die Laufzeit-APIs verwendet.

**SQL-Zugriff:** Auf alle Objekte in der Datenbank kann über das JDBC-API von Jalapeño automatisch auch mit SQL zugegriffen werden – ohne dass der Entwickler dafür ein objekt-relacionales Mapping vornehmen muss.

**Datenbank- und Plattformunabhängigkeit:** Caché ist für jede wichtige Plattform verfügbar. Wenngleich Jalapeño am besten mit Caché funktioniert, kann es mit Hilfe der Standard-DDL das zugrunde liegende Datenbankschema in ein entsprechendes relationales Schema exportieren. So kann eine Anwendung, die für den Objektzugriff mit Caché erstellt wurde, ebenso auf einer relationalen Datenbank betrieben werden.

## VORTEILE VON CACHE

### Schnelle Datenbereitstellung:

Web-Anwendungen, die Caché als Datenserver einsetzen, profitieren von der hohen Performance und der massiven Skalierbarkeit der multi-dimensionalen Daten-Engine von Caché.

### Schnellere .NET-Entwicklung:

Die Entwickler werden produktiver, wenn sie mit ihren bevorzugten Werkzeugen in vertrauter Umgebung arbeiten können. Durch die Bereitstellung des Datenzugriffs sowohl über SQL als auch über Objekte unterstützt Caché einen Großteil der gängigen Entwicklungstechniken und -werkzeuge.

## CACHÉ UND .NET

Dank seines offenen und flexiblen Datenzugriffs arbeitet Caché nahtlos mit .NET zusammen. Es gibt viele Möglichkeiten, die beiden Systeme zu verbinden, SQL, XML und SOAP eingeschlossen. Entwickler können Anwendungen mit den von ihnen bevorzugten Technologien erstellen – die alle von der überlegenen Performance und Skalierbarkeit von Caché profitieren.

### ADO.NET

ADO.NET ist die neue Inkarnation von ADO, die für den Einsatz im .NET-Framework optimiert ist. Es soll .NET-Anwendungen „datenbankunabhängig“ machen und verwendet in der Regel SQL zur Kommunikation mit Datenbanken. Über den relationalen Datenzugriff unterstützt Caché ADO.NET auf native Weise. Ebenso unterstützt es ODBC.NET von Microsoft und die Read-only SOAP-Konnektivität, die in ADO.NET integriert ist.

### Web Services

.NET kennt zwei Möglichkeiten, Web Services zu nutzen. Die eine besteht darin, XML-Dokumente über HTTP zu senden. Die andere Möglichkeit ist, das SOAP-Protokoll zur Vereinfachung des Austauschs von XML-Dokumenten zu nutzen. Caché unterstützt beide Darstellungsformen für Daten und kann somit nahtlos mit den Web Services von .NET zusammenarbeiten.

### Caché Managed Objects

Caché kann automatisch .NET-Assemblies (oder C#-Quellcode) aus Caché-Klassen erstellen. Ein Plug-in für Visual Studio ermöglicht Entwicklern, die diese Umgebung bevorzugen, den einfachen Zugriff auf Caché-Objekte.

## CACHÉ UND XML

Ähnlich wie HTML eine Internet-kompatible Markup-Sprache zur Anzeige von Daten in einem Browser ist, ist XML eine Markup-Sprache für den Austausch von Daten zwischen Anwendungen. Die Struktur von XML-Daten ist hierarchisch und multidimensional, so dass sie sich ideal zum Einsatz mit der multidimensionalen Daten-Engine von Caché eignet.

### Export von XML

Damit eine Caché-Klasse kompatibel zu XML wird, muss sie von der Klasse „XMLAdaptor“ erben, die in Caché enthalten ist. Diese stellt alle Methoden bereit, die benötigt werden, um:

- entweder eine DTD (Document Type Definition) oder ein XML-Schema für die Klasse zu erstellen. DTDs und Schemata werden von Caché automatisch erzeugt. Die XML-Darstellung einer Klasse lässt sich aber auch vom Entwickler anpassen.
- Daten (Instanzen der Klasse) automatisch gemäß der definierten DTD oder des Schemas als XML zu formatieren.

### Import von XML

Caché enthält auch Klassen, mit deren Methoden Entwickler:

- XML-Schemata importieren und automatisch die entsprechenden Caché-Klassen erstellen können.
- über ein einfaches API die in XML-Dokumenten enthaltenen Daten als Instanzen (Objekte) von Caché-Klassen importieren können.
- XML-Dokumente mit dem integrierten XML-(SAX-)Parser parsen und validieren können.

## CACHÉ UND WEB SERVICES

Web Services ermöglichen die gemeinsame Nutzung von Anwendungsfunktionalität über das Internet, aber auch innerhalb einer Organisation oder eines Systems. Web Services verfügen über eine Schnittstelle, die in WSDL (Web Service Definition Language) beschrieben ist und liefern ein XML-Dokument, das gemäß dem SOAP-Protokoll formatiert ist.

Mit Caché kann jede Klassenmethode, jede SQL Stored Procedure und jede beliebige Abfrage automatisch als Web Service dargestellt werden. Caché erzeugt die erforderliche WSDL-Beschreibung für den Service und sendet bei Aufruf des Service das entsprechend formatierte XML-Dokument. Darüber hinaus vereinfacht Caché die schnelle Anwendungsentwicklung, indem automatisch eine Webseite erzeugt wird, um den Service zu testen, ohne dass eine Client-Anwendung erstellt werden muss.

### VORTEILE VON CACHÉ

**Einfache Konnektivität zu XML:** Caché nutzt Mehrfachvererbung zur Bereitstellung einer bidirektionalen Schnittstelle zu XML für jede beliebige Caché-Klasse. Das Ergebnis: Caché-Klassen können einfach und schnell in XML-Dokumente und -Schemata umgewandelt werden. Ebenso können XML-Schemata und -Dokumente in Caché-Klassendefinitionen und Objekte umgewandelt werden.

**Beschleunigte Entwicklung schneller XML-Anwendungen:** Da die nativen multidimensionalen Datenstrukturen von Caché sehr gut zu XML-Dokumenten passen, müssen Entwickler keine manuelle Übersetzung zwischen XML und der Datenbank codieren. Und der geringere Verarbeitungsaufwand sorgt für einen schnelleren Ablauf der Anwendungen.

**Web Services auf Knopfdruck:** Jede Caché-Methode kann mit wenigen Mausklicks auch als Web Service veröffentlicht werden. Caché erzeugt automatisch die WSDL-Beschreibung und die SOAP-Antwort, wenn der Service aufgerufen wird.

## CACHÉ UND MULTIVALUE

Caché enthält die gesamte Funktionalität, die zur Entwicklung und Ausführung von MultiValue-Anwendungen (manchmal auch als Pick-Anwendungen bezeichnet) erforderlich ist, einschließlich den MultiValue-Elementen:

- Sprache MVBasic
- Dateizugriff
- Abfragesprache
- Data Dictionary
- “Procs”
- Command Shell

Diese MultiValue-Funktionalität ist ein integraler Bestandteil von Caché und keine separate MultiValue-Implementierung. Sie setzt auf der multidimensionalen Daten-Engine, der Laufzeitfunktionalität und den Entwicklungstechnologien von Caché auf. Dies bedeutet, dass MultiValue-Benutzer die gesamte Funktionalität von Caché hundertprozentig nutzen können.

### MultiValue-Dateizugriff

MultiValue-Anwendungen betrachten die Datenbank als eine Gruppe von Dateien, auf die über Lese- und Schreib-Operationen und über MultiValue-Abfragen zugegriffen werden kann. In Caché wird jede MultiValue-Datei als multidimensionale “Global”-Struktur gespeichert, wobei jeder Datensatz ein Knoten des Globals ist. Diese Funktion basiert auf der Fähigkeit von Caché, mehrere Datenelemente pro Knoten zu speichern.

Eine MultiValue-Datei, die Rechnungen speichert, kann beispielsweise die folgende Struktur aufweisen:

<b>Rechnung #</b>	<b>Artikel-ID</b>
<b>Kunde</b>	<b>Attribut 1</b>
<b>Rechnungsdatum</b>	<b>Attribut 2</b>
<b>Artikel</b>	<b>Attribut 3 (MultiValued)</b>
<b>Mengen</b>	<b>Attribut 4 (MultiValued)</b>
<b>Preise</b>	<b>Attribut 5 (MultiValued)</b>
...	...

Caché stellt diese MultiValue-Datei intern als multidimensionale Global-Struktur dar:

```
^Rechnung(Rechnung #) =  
Kunde ^ Rechnungsdatum ^ ArtikelNr1 ] ArtikelNr2 ^ Mengel ] Menge2 ^ Preis1 ] Preis2
```

wobei „^” das normale Attributbegrenzungszeichen angibt (ASCII 254) und “]” das Unterattribut-Begrenzungszeichen (ASCII 253).

Auf MultiValue-Dateien kann mit MultiValue-Programmen über normale READ/WRITE-Befehle und MultiValue-Abfragen zugegriffen werden. Auf sie kann über MVBasic und ebenso die anderen Sprachen zugegriffen werden. Alle normalen Mechanismen von Caché stehen zur Verfügung – einschließlich dem objektorientierten Zugriff, dem direkten Zugriff über multidimensionale Arrays und dem SQL-Zugriff.

MultiValue-Dateien können mit einer Vielzahl an Sortierfolgen und Indextypen indiziert werden (z.B. auch Bitmap-Indizes).

## MultiValue Query Language

Die MultiValue Query Language sorgt sowohl für die Datenauswahl als auch die Listenformatierung für MultiValue-Dateien. Die Abfragesprache kann in MVBasic, in der Command Shell und in „Procs“ verwendet werden. Caché übersetzt MultiValue-Abfragen in Caché SQL-Abfragen mit zusätzlichem Code zur Unterstützung der Korrelate, der Listenformatierung und der verschiedenen anderen zusätzlichen Funktionen. Da diese Abfragen von der hochperformanten SQL-Engine von Caché ausgeführt werden, wird die Zuverlässigkeit erhöht, die Ausführung optimiert und es stehen sämtliche ausgeklügelten Indexmechanismen von Caché zur Verfügung. Natürlich können MultiValue-Entwickler auch direkt Caché SQL verwenden, wenn dies gewünscht wird.

## MultiValue Data Dictionary

Eine MultiValue-Datei kann eine entsprechende Dateibeschreibung im MultiValue Data Dictionary haben, die über MVBasic-Code und den gewohnten MultiValue “ED”-Editor direkt bearbeitet werden kann. Die MultiValue Query Language verwendet dieses Dictionary.

Jede MultiValue-Datei kann auch über eine entsprechende Caché-Klassendefinition verfügen. Nötig ist eine solche Klassendefinition, wenn mit den Daten über den Objekt- oder SQL-Zugriff gearbeitet werden soll (zur Verwendung der MultiValue Query Language wird sie nicht gebraucht). Eine Klassendefinition ist auch erforderlich, wenn Indizes angelegt werden sollen.

Beim Import älterer MultiValue-Anwendungen können die Caché-Klassendefinitionen automatisch aus dem MultiValue-Dictionary erstellt werden. Wird ein SQL- oder objektorientierter Zugriff gewünscht, muss die daraus resultierende Klasse in den meisten Fällen allerdings noch bearbeitet werden, um die Daten mit mehr Bedeutung zu versehen. Ein Studio Wizard hilft bei der Automatisierung der Klassenerstellung aus MultiValue-Dictionaries und ermöglicht es auch, zu einem späteren Zeitpunkt weitere Mappings vorzunehmen. Standardmäßig sind diese Klassen Read-Only, d.h. die Daten können über Objekte und SQL nur gelesen, nicht aber aktualisiert werden, und werden unabhängig vom MultiValue-Dictionary editiert.

Beim Erstellen einer neuen MultiValue-Datei empfiehlt es sich, zuerst eine Caché-Klasse zu erstellen, die von der MVAdaptor-Oberklasse erbt. Hierdurch wird ein MultiValue-kompatibles Dateiformat für die Datenspeicherung verwendet und automatisch eine Dateibeschreibung im MultiValue Data Dictionary erstellt. Die Daten der Datei sind dann über alle Caché-Zugriffsmechanismen zugreifbar und aktualisierbar, einschließlich dem objektorientierten Zugriff, dem Zugriff über SQL und dem direkten multidimensionalen Global-Zugriff. Ab dann sollte eher die Caché-Klassendefinition als das MultiValue-Dictionary bearbeitet werden, da Bearbeitungen der Klasse automatisch ins MultiValue-Dictionary übernommen werden.

## MultiValue und Objekte

In MVBasic können Objekte genauso wie in Caché Basic verwendet werden, mit der Ausnahme, dass MVBasic den Objektzugriff anstatt der Punktyntax durch die Syntax “->” darstellt. Jede Klasse aus dem Class Dictionary kann verwendet werden, unabhängig davon, in welcher Sprache die Methoden der Klasse erstellt wurden.

In den folgenden Beispielen ist “person” eine Objektreferenz:

```
person->Name = "Schmid, Hans" // Legt den Namen der Person fest.
person->Privatanschrift->Stadt // Gibt den Wohnort der Person an.
person->Arbeitgeber->Name     // Holt das Objekt „Arbeitgeber“ der Person in
                             // den Speicher
                             // und greift auf den Namen des Arbeitgebers zu.
person->Save()                // Speichert die Person auf Platte.
```

## Command Shell von MultiValue

Die MultiValue Command Shell kann in einer Terminal-Umgebung ausgeführt werden. Zusätzlich zu den normalen MultiValue Command Shell-Funktionen ermöglicht Caché, dass MVBasic-Befehle direkt in der Command Shell ausgeführt werden. Zum Beispiel ergibt die Eingabe:

```
:: DIM A(34)
:: FOR I = 1 TO 34 ; A(I) = I; NEXT
:: FOR I = 1 TO 34 ; CRT A(I):" "; ; NEXT
```

folgendes Ergebnis:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
```

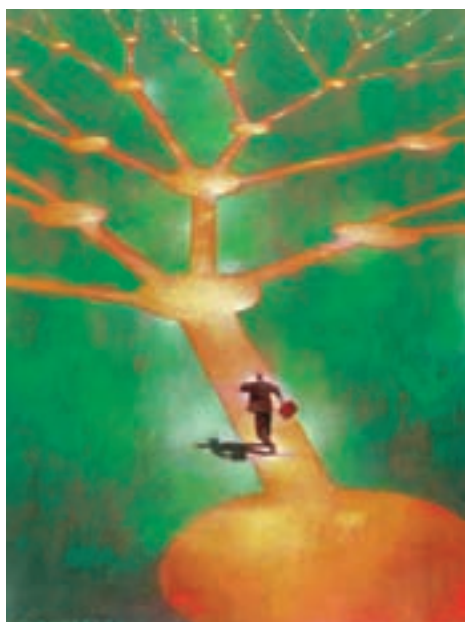
## VORTEILE VON CACHÉ

**Neues Leben für alte Anwendungen:** Mit Caché ist es ein Leichtes, ältere MultiValue-Anwendungen mit Browser-basierten Benutzeroberflächen, objektorientiertem Zugriff, robustem SQL oder Web Services zu modernisieren. MultiValue-Anwendungen erhalten eine stabile Basis durch eine moderne Datenbank, die in heutigen anspruchsvollen Umgebungen gut akzeptiert ist und ständig weiterentwickelt wird.

**Neue Anwendungen schnell erstellen:** Da MultiValue in Caché als zusätzliche Skriptsprache und Dateizugriff implementiert ist, stehen alle nativen Funktionen von Caché zur Verfügung, um schnell neue Funktionalität zu definieren. MultiValue-Programmierer können die Vorteile der objektorientierten Programmierung nutzen und einfach mit anderen Anwendungen interagieren, während sie weiterhin mit MVBasic arbeiten.

**Hohe Performance und Skalierbarkeit mit außergewöhnlicher Zuverlässigkeit für MultiValue-Anwender:** Caché bietet MultiValue-Anwendern eine signifikant höhere Performance und Skalierbarkeit. Darüber hinaus ist Caché für kritische Umgebungen ausgelegt, bei denen die Verfügbarkeit rund um die Uhr gewährleistet sein muss und Ausfälle nicht toleriert werden. Caché bietet ausgefeiltes Journaling, Transaktionsverarbeitung, eine ausfallsichere Datenbank und fehlertolerante Konfigurationen.

## Kapitel 4: Schnelle Web-Anwendungen schnell entwickelt mit Caché Server Pages (CSP)



Der Slogan „Schnelle Web-Anwendungen schnell entwickelt“ enthält gleich zweimal das Wort schnell. Dies rührt ganz einfach daher, dass es mit Caché Server Pages möglich ist, anspruchsvolle, datenbankorientierte Web-Anwendungen schneller zu entwickeln als mit herkömmlichen Methoden. Zudem ist die Caché-Datenbank die schnellste weltweit und es lassen sich damit Systeme mit Zehntausenden von gleichzeitigen Benutzern realisieren.

Es gibt vielerlei Möglichkeiten, mit Caché eine Web-Anwendung zu entwickeln – einschließlich der traditionellen Methode des SQL-Zugriffs auf die Datenbank. In diesem Kapitel beschäftigen wir uns aber mit einem anderen, weitaus direkteren Ansatz: den Caché Server Pages (CSP).

CSP ist eine Technologie, die zum Applikationsserver von Caché gehört. Es ist für Caché-Anwendungen die schnellste Möglichkeit, mit dem Web zu interagieren. Sie bietet folgende Leistungsmerkmale:

- einen hochentwickelten objektorientierten Entwicklungsansatz
- extrem hohe Performance und Skalierbarkeit zur Laufzeit

CSP unterstützt HTML, XML und andere Web-orientierte Markup-Sprachen.

CSP ist kein Web-Design-Tool, obwohl es mit solchen zusammen verwendet werden kann. Während Web-Tools sich häufig auf die Entwicklung von statischen HTML-Seiten ausrichten, geht CSP über die Seitengestaltung hinaus und dient vor allem der Entwicklung der Anwendungslogik. Zudem stellt CSP eine Laufzeit-Umgebung zur Verfügung, die die schnelle Ausführung von Code mit dem Caché-Applikationsserver ermöglicht.

CSP unterstützt eine mächtige prozedurale Programmierumgebung, mit der weitaus spezialisiertere und anspruchsvollere Applikationen entwickelt werden können als das mit reinen Anwendungsgeneratoren möglich ist. Es unterstützt die schnelle Entwicklung zum einem durch die Klassenarchitektur, durch die miteinander kombinierbare „Code-Bausteine“ erstellt werden können, und zum anderen durch Wizards, die schnell einfache Versionen von anpassbarem Code erzeugen können. Das Ergebnis sind schnell entwickelte, ausgefeilte Web-Anwendungen.

### VORTEILE VON CACHÉ

Objektorientierte Programmierung und Caché Wizards führen zu einer schnellen Entwicklung von anspruchsvollen, browserbasierten Datenbank Anwendungen.

Die Caché Server Pages zeichnen sich unter anderem durch folgende Funktionen aus:

- **Dynamic Server Pages** – Da die Seiten dynamisch auf dem Applikationsserver durch Anwendungscode entstehen, anstatt dass ein Web-Server statisches HTML zurückgibt, können Anwendungen schnell auf eine große Zahl verschiedener Anforderungen reagieren und die Ergebnisseiten individuell auf die Anfragen zuschneiden.
- **Session-Modell** – Die gesamte Verarbeitung für alle Seiten aus einem Browser wird als Teil einer Session aufgefasst – von der ersten Browser-Anforderung bis die Anwendung abgeschlossen (oder ein Timeout aufgetreten) ist.
- **Serverseitige Zustandsspeicherung** – Innerhalb einer Session können Anwendungsdaten – bis hin zum vollständigen Applikationskontext – über verschiedene Browser-Anforderungen hinweg auf dem Server zwischengespeichert werden. Dies vereinfacht die Entwicklung und Ausführung komplexer Anwendungen.
- **Objektarchitektur** – Da jede Seite einer Klasse entspricht, können Code und andere gemeinsame Eigenschaften von Seiten durch Vererbung einfach vereinheitlicht werden. Typischerweise werden auch die Daten über Objekte referenziert, mit allen Vorteilen objektorientierter Programmierung.
- **XML** – XML ist eine leistungsstarke Alternative zu HTML für der Erstellung von Webseiten. CSP funktioniert mit XML genauso wie mit HTML. Die Page()-Methode liefert dann anstelle von HTML ganz einfach XML zurück.
- **Caché Application Tags zur automatischen Generierung von Anwendungscode auf dem Server** – Diese erweiterten HTML-Tags sind genauso einfach einzusetzen wie herkömmliche HTML-Tags. Fügt man sie in eine HTML-Datei ein, generieren sie ausgefüllten Anwendungscode, der eine Vielzahl von Funktionen umfassen kann: Öffnen von Objekten, Ausführen von Abfragen, Programmflusskontrolle usw. Diese Tags sind erweiterbar. Entwickler können eigene Tags erstellen und so spezielle Anforderungen abdecken.
- **Einbindung in bekannte Web-Tools** – CSP funktioniert mit einer Vielzahl von Tools für die visuelle Gestaltung von Webseiten. Für Macromedia Dreamweaver geht CSP sogar noch einen Schritt weiter, dort können Caché Application Tags einfach per Mausklick eingebaut werden. CSP umfasst auch einen entsprechenden Wizard, der einfache Formulare zur Dateneingabe und -bearbeitung erzeugt.
- **Serverseitige Methoden, die von Browser-Anwendungen aufgerufen werden** – Um die Entwicklung von dynamischen und interaktiven Anwendungen zu unterstützen, vereinfacht CSP den Aufruf von serverseitigen Methoden. Bei Ereignissen auf der Browserseite – in der Regel, wenn ein Benutzer eine Aktion ausgeführt hat – kann auf dem Server automatisch Anwendungscode aufgerufen und eine Antwort darauf erstellt werden, ohne dass erst eine ganze neue Seite übertragen und angezeigt werden muss.
- **Verschlüsselung** – Caché verschlüsselt Daten im URL automatisch, um die Berechtigung von Anforderungen zu prüfen und Fälschungen zu verhindern. Der Schlüssel wird ausschließlich auf dem Server verwaltet, wo er nur für eine einzige Session verwendet wird.

*Das alles ist viel Technologie – aber sie muss deshalb nicht schwierig einzusetzen sein. Wir haben Wert darauf gelegt, CSP einfach zu machen. Die meisten dieser Funktionen werden automatisch für Sie ausgeführt. Unsere Philosophie lässt sich auf einen einfachen Nenner bringen: „Power through Simplicity“ – die Komplexität soll in unserer Implementierung und nicht in Ihrer Programmierung liegen.*

## DAS SERVER PAGE-MODELL VON CACHÉ

### Traditionelle Web-Technologien

Bei herkömmlichen Web-Technologien wird eine Anforderung an den Web-Server gesendet, der dann eine HTML-Datei abrufen und zurück an den Browser schicken. Sobald variable Daten ins Spiel kommen, wird die Entwicklung um einiges schwieriger. Meist wird dann auf dem Web-Server CGI (typischerweise mit Sprachen wie Perl oder Tcl) eingesetzt, um SQL-Abfragen und Stored-Procedure-Aufrufe an die Datenbank zu senden. Eine solche Programmierumgebung lässt in vielen Punkten zu wünschen übrig, und auch der Betrieb – vor allem bei einer großen Zahl von Benutzern – wird schnell ineffizient, da der Web-Server massiv belastet wird.

Bei CGI löst jede Browser-Anforderung normalerweise einen neuen Prozess aus. Um diesen Mehraufwand zu umgehen, wird manchmal Anwendungs-Code direkt mit dem Web-Server gelinkt, was den unglücklichen Nebeneffekt hat, dass Fehler in der Anwendung zum Crash des gesamten Web-Servers führen können.

### Dynamic Server Pages

CSP verfolgt einen anderen Ansatz zur Programmierung und Ausführung: Dynamic Server Pages. Hier werden Inhalte (HTML, XML, Style Sheets, Bilder und so weiter) zur Laufzeit automatisch auf dem Applikationsserver von Caché generiert, statt sie fertig aus Dateien abzurufen. Dies führt zu wesentlich mehr Flexibilität bei der Beantwortung von Seitenanforderungen.

Der größte Teil des Anwendungscodes wird direkt auf dem Caché-Applikationsserver ausgeführt, welcher sich auf demselben Rechner wie der Web-Server befinden kann, aber nicht muss. Etwas Code – in der Regel JavaScript oder Java – wird im Browser ausgeführt, zum Beispiel für einfache Datenvalidierungen, Umformatierungen oder den Aufruf von serverseitigem Code.

Auf diesem Weg kann sichergestellt werden, dass nicht für jede Browser-Anforderung ein neuer Prozess gestartet werden muss (wie es beim traditionellen CGI-Ansatz üblich ist), so dass die Performance enorm steigt. Und da der Anwendungscode nicht mit dem Web-Server verknüpft ist, kann ein Anwendungsfehler unmöglich den Web-Server zum Absturz bringen.

### Sessions – das Verarbeitungsmodell

Die gesamte Verarbeitung für alle Seiten aus einem Browser wird als Teil einer Session aufgefasst – von der ersten Browser-Anforderung bis die Anwendung abgeschlossen oder der programmierbare Timeout abgelaufen ist. Wenn der Web-Server eine Seitenanforderung (URL) mit der Endung „.csp“ empfängt, wird diese Anforderung (mit Hilfe einer dünnen Schicht Caché-Code auf dem Web-Server) dem zuständigen Caché-Applikationsserver weitergereicht, der sich auch auf einem anderen Computer befinden kann.

Wenn der Caché-Applikationsserver die Anforderung erhält, prüft er zunächst, ob bereits eine Session für diesen Browser aktiv ist. Ist dies nicht der Fall, wird automatisch eine gestartet. Caché führt dann Anwendungscode aus, der direkt zur entsprechenden Seite gehört. Dabei werden die vom Benutzer geforderten Aktionen ausgeführt und automatisch HTML-Code, XML-Code, Bilder oder andere Inhalte erzeugt, die zurück an den Browser geschickt werden.

Eine Session endet durch das Setzen einer entsprechenden Eigenschaft im Session-Objekt. Anwendungen, die zustandslos ausgeführt werden, bevorzugen unter Umständen, die Session mit jeder Seite zu beenden.

## Zustandserhaltung

Eine der großen Herausforderungen für Programmierer ist die inhärent zustandslose Natur des Webs – es besteht in der Regel keine Möglichkeit, Informationen zwischen zwei Anforderungen auf dem Server zu behalten. Stattdessen senden Anwendungen jedes Mal alle nötigen Zustandsinformationen als Teil des URL oder in versteckten Formularfeldern erneut an den Browser. Dies ist keine effiziente Technik für komplexe Applikationen, weshalb Daten dann oft temporär in Dateien oder Datenbanken zwischengespeichert werden. Das führt dann jedoch zwangsläufig zu erheblichem Overhead auf dem Server und erschwert die Programmierung.

Das Session-Modell von Caché hingegen erlaubt es, die von der Anwendung benötigten Daten zwischen den einzelnen Browser-Anfragen automatisch und effizient zu erhalten. Dazu verwendet CSP ein Session-Objekt, das allgemeine Session-Informationen sowie Eigenschaften enthält, mit denen Programmierer verschiedenste Charakteristika der Session kontrollieren können. In diesem Session-Objekt kann die Anwendung dann ihre eigenen Daten speichern, die automatisch von einer Anfrage zur nächsten erhalten bleiben sollen.

Die Anwendung legt auch fest, wie viel von der Zustandsinformation erhalten bleiben soll. Dies erfolgt durch Setzen der Eigenschaft „Preserve“ des Session-Objekts auf 0 oder 1 (die Standardeinstellung ist 0, sie kann aber jederzeit zur Laufzeit geändert werden).

**0** – Die Daten, die im Session-Objekt enthalten sind, werden beibehalten. (Diese Daten werden einfach in einer multidimensionalen Eigenschaft abgelegt, die ohne vorherige Deklaration Daten beliebigen Typs sowie eine beliebige Anzahl von Subscripts aufnimmt.)

**1** – Caché teilt jeder Session einen dedizierten Prozess zu, so dass der gesamte Prozesszustand erhalten bleibt, inklusive aller Variablen (also nicht nur jener aus dem Session-Objekt), geöffneter Geräte und gesetzter Sperren.

Die Einstellung 0 erlaubt die logische Partitionierung aller beibehaltenen Daten und ermöglicht, dass mehrere Sessions sich einen Server-Prozess teilen, behält jedoch weniger Zustandsinformationen bei. Die Einstellung 1 ist einfacher für den Entwickler und stellt mehr Möglichkeiten zur Verfügung, führt aber zu einem etwas höheren Ressourcenverbrauch.

## Das Request-Objekt

Neben dem Session-Objekt stellt CSP automatisch noch weitere Objekte zur Verfügung, die dem Programmierer die Verarbeitung der Seite erleichtern. Eines davon ist das Request-Objekt. Beim Empfang einer Seite werden der URL dekodiert und die Inhalte im Request-Objekt abgelegt. Das Request-Objekt enthält alle Namen/Wert-Paare und Formulardaten sowie andere nützliche Informationen. So kann beispielsweise der Wert des Namens „FilmID“ durch folgenden Code erhalten werden:

```
%request.Data("FilmID",1) // die 1 gibt an, dass wir den ersten
                          Wert für diese Namen möchten,
                          // falls mehrere Werte hierfür
                          vorhanden sind.
```

## VORTEILE VON CACHÉ

Durch die automatische Bewahrung der Zustandsinformationen auf dem Server fallen weniger Netzwerkverkehr und geringerer Overhead auf dem Server an, da die Anwendung nicht bei jeder Seitenanforderung auf Daten zugreifen und diese speichern muss. Darüber hinaus vereinfacht dies die Programmierung der Anwendung erheblich.

Die Verwendung von Dynamic Server Pages und des Caché-Applikationsservers führt zu einer größeren Flexibilität bei der Beantwortung von Web-Anforderungen sowie einer komplexeren Programmierumgebung ohne das Risiko, dass Anwendungsfehler den Web-Server lahm legen.

## DIE KLASSENARCHITEKTUR VON WEBSEITEN

Für jede Webseite existiert eine korrespondierende Klasse, die die Methoden (Code) enthält, welche den Seiteninhalt generieren. Geht eine Anforderung ein, wird anhand des URL die entsprechende Seitenklasse ermittelt und die Methode „Page()“ dieser Klasse aufgerufen. Normalerweise sind Seitenklassen von der Standard-Webseitenklasse %CSP.Page abgeleitet, die verschiedene integrierte Funktionen wie zum Beispiel die Erstellung von Headern oder Codierungen zur Verfügung stellt. Die Standardvorgaben können auf verschiedene Art und Weise überschrieben werden: Durch Ableitung von einer anderen Oberklasse, durch Mehrfachvererbung oder einfach durch das Überschreiben einzelner Methoden.

Diese Klassenarchitektur macht es einfach, das Verhalten einer ganzen Anwendung zu verändern und einen einheitlichen Stil durchzusetzen. Sie macht außerdem alle weiteren Vorteile der objektorientierten Programmierung auch für die Web-Entwicklung verfügbar.

Die Seitenklasse enthält den Code, um die gewünschte Anforderung auszuführen sowie eine Antwort zu generieren und an den Browser zurückzusenden. Aber nicht der gesamte Code, der ausgeführt wird, befindet sich in dieser Seitenklasse. Üblicherweise findet sich der meiste ausgeführte Code in Methoden verschiedener Datenbankklassen und eventuell in zusätzlichen Geschäftslogik-Klassen. So besteht der Entwicklungsprozess aus der Entwicklung sowohl von Seitenklassen als auch von Datenbankklassen (und eventuell zusätzlichen Geschäftslogik-Klassen).

Im Allgemeinen empfiehlt sich, in Seitenklassen nur die Benutzeroberflächenlogik abzulegen. Geschäfts- und Datenbanklogik sollten in anderen Klassen abgelegt werden, so dass eine klare Trennung zwischen dem Code für die Benutzeroberfläche und der Geschäfts- und Datenbanklogik besteht und weitere Benutzeroberflächen zu einem späteren Zeitpunkt einfach hinzugefügt werden können.

## MEHRERE ENTWICKLUNGSSTRATEGIEN

Für jede Webseite wird eine Seitenklasse angelegt, die den Code enthält, der für diese Seite ausgeführt werden soll. Es bestehen verschiedene Möglichkeiten, Seitenklassen zu erstellen, wobei die meisten Anwendungen mehrere nutzen:

- **CSP-Datei** – Eine HTML-Datei mit eingebetteten Caché Application Tags wird mit einem einfachen Texteditor oder einem Web-Design-Werkzeug erstellt. Diese sequenzielle Datei (eine „CSP-Datei“) wird nicht direkt an den Browser gesendet – vielmehr wird sie zunächst intern zu einer Seitenklasse kompiliert.
- **Direkte Programmierung** – Programmierer können die vollständige Seitenklasse selbst schreiben, indem sie die entsprechenden Methoden kodieren.



## CSP-DATEIEN

CSP-Dateien sind spezielle HTML-Dateien mit eingebetteten Caché Application Tags, aus denen Seitenklassen kompiliert werden – die gleiche Art von Seitenklassen, die ein Programmierer auch direkt erstellen kann. Diese Seitenklassen werden dann kompiliert, um Code zu erzeugen, der auf dem Caché-Applikationsserver zur Beantwortung der Client-Anforderungen läuft.

Caché Studio enthält einen Form Wizard, der automatisch eine CSP-Datei erzeugt, mit der man eine Datenbankklasse bearbeiten oder anzeigen kann. Der Benutzer klickt auf die betreffende Datenbankklasse und dann auf die Eigenschaften, die verwendet werden sollen. Der Caché-Wizard erledigt den Rest: Er fügt HTML-Code und die Caché Application Tags zu der Seite hinzu. Da der Wizard HTML ausgibt, können Sie das Ergebnis einfach bearbeiten, wenn es noch nicht ganz Ihren Vorstellungen entspricht.

Der CSP-Datei-Ansatz ist aus den folgenden Gründen sehr leistungsstark:

- Web-Designer können das grafische Layout entwerfen, während sich die Programmierer auf den Code konzentrieren.
- Ein Großteil der Benutzeroberfläche kann nicht-prozedural in einer grafischen Umgebung programmiert und von Geschäfts- und Datenbanklogik getrennt gehalten werden.
- Es ist oft einfacher, eine Anwendung für einen bestimmten Benutzer anzupassen, indem Nicht-Programmierer die grafische Darstellung bearbeiten und einfache Funktionen hinzufügen.

Da die grafischen Spezifikationen der Anwendung von dem Großteil der Programmierlogik getrennt sind, lässt sich das Erscheinungsbild relativ einfach ohne Neuprogrammierung verändern. Man muss nur die HTML- oder XML-Datei bearbeiten und die Seite rekompilieren.

Obwohl auf diese Weise eine einfache Anwendung erstellt werden kann, fügen Programmierer in der Regel zusätzlichen Code hinzu. Dieser zusätzliche Code wird über Application Tags bereitgestellt, die entweder prozeduralen Code enthalten oder Code in anderen Klassen aufrufen. Oftmals ist es einfacher, komplexe Seiten mit viel prozeduralem Code direkt zu programmieren, anstatt eine CSP-Datei zu verwenden.

Caché beinhaltet auch ein Add-in für Dreamweaver, ein beliebtes Werkzeug für das Design von Webseiten. Caché Application Tags können ganz einfach per Mausklick hinzugefügt werden. Ebenso ist ein Caché Form Wizard für Dreamweaver enthalten, der automatisch den Code erzeugt, der für die Anzeige oder die Bearbeitung von Datenbankobjekten erforderlich ist.

### Caché Application Tags

Caché Application Tags können in CSP-Dateien eingebettet werden. Sie werden wie HTML-Standard-Tags verwendet, sind aber in Wirklichkeit Anweisungen an den Caché Web Compiler, Anwendungs-Code zu generieren, der verschiedenste Funktionen zur Verfügung stellt. Hierzu gehören der Zugriff auf Datenbankobjekte, die Ausführung von Abfragen, die Programmflusskontrolle und die Ausführung des Codes auf dem Applikationsserver. Caché Application Tags sind erweiterbar. Entwickler können ihre eigenen Tags erstellen, um ihre Entwicklungsumgebung exakt auf ihre Anforderungen abzustimmen.

Caché Application Tags sind nicht in dem HTML-Code enthalten, der später an den Browser gesendet wird – sie befinden sich nur in der CSP-Datei, die vom Caché Web Compiler gelesen wird. Der Compiler wandelt sie automatisch in Standard-HTML um, das von jedem Browser interpretiert werden kann.

## HYPER-EVENTS

Mit den zu CSP gehörenden Hyper-Events können Ereignisse, die im Browser stattfinden (Mausklicks, Änderung von Feldwerten, Timeouts etc.), serverseitige Methoden auslösen und die angezeigte Seite aktualisieren, ohne dass diese neu aufgebaut werden muss. Nachdem die entsprechende Aktion ausgeführt wurde, kann die Server-Methode Code – in der Regel JavaScript – zur Ausführung im Browser zurückgeben. Mit Hyper-Events werden Web-Anwendungen interaktiver und kommunikativer.

Innerhalb einer CSP-Seite wird eine serverseitige Methode einfach mit folgender Syntax aufgerufen:

```
"#server(...)#"
```

Ein Beispiel: Wenn ein Benutzer auf eine Warenkorb-Grafik klickt, soll eine Server-Methode namens **InWarenkorb()** aufgerufen werden. Die HTML-Definition für die Abbildung könnte Folgendes enthalten:

```
onClick="#server(..InWarenkorb())#"
```

Der Web-Compiler ersetzt diese Syntax durch den JavaScript-Code, der bei Ausführung im Browser die Cache-Server-Methode aufruft.

## ZEN UND KOMPONENTENBASIERTE WEBSEITEN

ZEN ermöglicht die schnelle und einfache Entwicklung komplexer und datenintensiver Webanwendungen, die über ein differenziertes optisches Erscheinungsbild und eine hochgradig interaktive Benutzeroberfläche verfügen. ZEN ist keine 4GL-Sprache – ZEN ist eine umfangreiche Bibliothek vordefinierter Objektkomponenten und Entwicklungstools auf Basis der CSP- und Objekttechnologie von InterSystems. Dabei eignet sich ZEN insbesondere für die Entwicklung webbasierter Client/Server-Anwendungen, als Alternative zu Tools wie Visual Basic oder PowerBuilder.

ZEN-Komponenten ermöglichen eine Vielzahl zusätzlicher dynamischer Interaktionen – der Transfer von Daten an einen Server beschränkt sich nicht auf Standard Submit-Verfahren. So können beispielsweise mit der ZEN-Formkomponente individuell angepasste Validitätsprüfungen durchgeführt werden, einschließlich serverseitiger Methodenaufrufe ohne Page Request und dem damit verbundenen Neuaufbau des Bildschirms. Dies ermöglicht dem Anwender eine natürliche Art der Dateneingabe, vergleichbar mit klassischen Desktop Anwendungen.

ZEN verwendet die Session Management-Mechanismen von CSP und unterstützt die Benutzerauthentifizierung, Datenverschlüsselung und das Beibehalten persistenter Sessioninhalte über Page Requests hinweg. Die gesamte Kommunikation zwischen Browser und Server findet mittels eines Austauschs von Objekten statt, wobei eine Weiterentwicklung des AJAX (Asynchronous JavaScript and XML) Mechanismus genutzt wird.

ZEN-basierte Webseiten können auf einfache Weise mit CSP-Seiten kombiniert werden.

### Was versteht man unter einer ZEN-Komponente?

Eine ZEN-Komponente ist eine Klassendefinition, die das Erscheinungsbild und das Verhalten der Komponente auf einer Webseite angibt. Die ZEN-Klassendefinition enthält dabei in einem einzigen Dokument die Definition der Komponente, einschließlich Stylesheets, sowie serverseitig und clientseitig auszuführendem Code.

Zur Laufzeit erstellt ZEN zwei Objekte für jede in der Webseite verwendete Komponente: ein Objekt auf der Client-Seite, das ZEN automatisch als JavaScript-Objekt im Browser anlegt, und ein Objekt auf der Server-Seite. ZEN verwaltet automatisch den Zustand beider Objekte, hält diese aktuell und verwaltet den Informationsfluss zwischen den Objekten.

## Arten von ZEN-Komponenten

Die ZEN-Bibliothek verfügt für alle Standard HTML Controls – wie Buttons, Eingabe- und Textfelder – über eine ZEN-Komponente, die diese implementiert. Diese Komponenten erben ihre grundlegenden Verhaltensweisen von der ZEN Control Klasse.

ZEN enthält darüber hinaus komplexere Komponenten, die automatisch Daten aus einer Datenbank anzeigen können und diese als Reaktion auf Benutzeraktionen dynamisch aktualisieren. Zum Beispiel zeigt die leistungsstarke Tabellenkomponente von ZEN unter Verwendung einer Datenbankabfrage automatisch Daten in einer HTML-Tabelle an. Die Tabellenkomponente unterstützt Paging, Scrollen, Sortieren nach Spalten, Filtern und eine Vielzahl an Darstellungsmöglichkeiten. Der Inhalt einer Tabelle kann dabei mit serverseitig gespeicherten Daten aktualisiert werden, ohne die gesamte Seite neu aufbauen zu müssen.

Weitere ZEN-Komponenten sind:

- **Menu** – ermöglicht die Darstellung verschiedener Menüarten
- **Grid** – fügt editierbare Gitter im Stil von Tabellenkalkulationen in eine Webseite ein
- **Tree** – zeigt hierarchische Daten in einer Baumstruktur an
- **Tab** – enthält eine Reihe von Registerkarten, von denen jede wiederum eine Reihe anderer ZEN-Komponenten enthalten kann
- **Chart** – ermöglicht die Darstellung von Diagrammen auf Basis von Standard Vector Graphics (SVG), wie beispielsweise Linien-, Flächen-, Balken-, Kuchen- und Streudiagramme
- **Graphical Meters** – enthält Geschwindigkeitsanzeiger, Messgeräte usw. um Daten als dynamische visuelle Komponenten (z. B. in Dashboards) anzuzeigen.

## Modifizierung des Erscheinungsbilds der ZEN-Komponenten

Alle ZEN-Komponenten verfügen über Eigenschaften zur Festlegung ihres ‚Look and Feels‘. Anwendungen können diese Eigenschaften zur Laufzeit dynamisch belegen, um dargestellte Werte, das Erscheinungsbild und das Verhalten der Komponenten zu ändern.

Das visuelle Erscheinungsbild wird durch CSS (Cascading Style Sheet) Definitionen gesteuert. Um eine individuelle Anpassung des Aussehens, beispielsweise durch angepasste Schriften, Farben und Größen, zu ermöglichen, können die CSS-Definitionen sowohl pro Komponente, webseitenweise, oder anwendungsweit überschrieben werden.

Für größere Modifikationen im Erscheinungsbild und Verhalten einer ZEN-Komponente können Subklassen der Komponente gebildet und entsprechend erweitert werden.

## Neue ZEN-Komponenten erstellen

Eine der großen Stärken von ZEN ist die einfache Erstellung neuer Komponenten.

Jede ZEN-Komponente ist als Klasse implementiert. Die Erstellung einer neuen Komponente erfolgt in vier Schritten: (1) eine neue Komponenten-Klasse wird erstellt, die Subklasse einer bereits vorhandenen Komponente sein kann; (2) der erstellten Klasse wird eine Methode hinzugefügt, die festlegt, was als HTML-Inhalt der Komponente dargestellt wird; (3) server- und clientseitige Methoden werden definiert, die das Verhalten der Komponente zur Laufzeit implementieren; (4) das Aussehen der neuen Komponente wird über die CSS-Definitionen festgelegt.

## Lokalisierung einer ZEN-Anwendung in verschiedenen Sprachen

ZEN kann automatisch die Menge aller Textwerte (Titel, Bildunterschriften usw.), die in den Komponenten einer Anwendung angezeigt werden, in einer speziellen Lokalisierungstabelle erfassen. Die Lokalisierungstabelle einer Anwendung kann dann als XML-Dokument exportiert werden, um die darin enthaltenen Textwerte in andere Sprachen zu übersetzen und anschließend als neue Tabelle wieder zu importieren.

Zur Laufzeit verwendet ZEN die Textwerte basierend auf der aktuellen Spracheinstellung im Browser des Anwenders.

## SVG-Unterstützung

SVG (Scalable Vector Graphics) sind ein leistungsstarker Standard, um grafische Daten auf einer Webseite anzuzeigen. In ZEN können grafische Komponenten mit Hilfe von SVG dargestellt werden. Dazu enthält die ZEN-Bibliothek eine umfangreiche Menge vordefinierter SVG-basierter Komponenten.

## Welche Browser unterstützt ZEN?

ZEN unterstützt Firefox (ab Version 1.5) und den Internet Explorer (ab Version 6.0). Für die Verwendung des Internet Explorers ist das Adobe SVG-Plug-In erforderlich, um die SVG-Komponenten von ZEN zu verwenden (in Firefox ist dieses standardmäßig enthalten). Die Unterschiede zwischen den verschiedenen SVG-Plug-Ins von Firefox und Internet Explorer werden von der ZEN-Bibliothek automatisch verwaltet.

### VORTEILE VON CACHE

**Umfangreiche Benutzerschnittstellen für Webanwendungen:** Es können visuell ausgefeilte, hochgradig interaktive Webseiten generiert werden, die den Benutzeroberflächen von Desktop-Anwendungen ähnlicher sind als herkömmliche einfache Browser-Formulare. Dieses interaktive Darstellungsformat führt zu hohem Bedienkomfort für den Benutzer.

**Schnelle objektbasierte Entwicklung:** Die Nutzung vordefinierter Komponenten beschleunigt die Entwicklung und vereinfacht spätere Modifikationen.

**Konsistente Benutzerschnittstellen:** Die komponentenbasierte Architektur vereinfacht die Definition und Durchsetzung eines anwendungsweit einheitlichen ‚Look and Feels‘.





**World Headquarters**

**InterSystems Corporation**

World Headquarters  
One Memorial Drive  
Cambridge, MA 02142  
USA

Tel: +1.617.621.0600

Fax: +1.617.494.1631

**[www.InterSystems.com](http://www.InterSystems.com)**

**Deutschland**

**InterSystems GmbH**

Hilpertstr. 20a  
D-64295 Darmstadt  
Tel.: +49.6151.1747-0  
Fax: +49.6151.1747-11

**[www.InterSystems.de](http://www.InterSystems.de)**

**Schweiz**

**InterSystems B.V.**

In der Luberzen 42  
CH-8902 Urdorf  
Tel.: +41.43.455.7711  
Fax: +41.43.455.7722

**[www.InterSystems.ch](http://www.InterSystems.ch)**

InterSystems  
**CACHE**