

Analyse von Enterprise Application Servern und  
Datenbank Management Systemen für  
FlexiTRUST

26. September 2002

Lutz Feldgen

Studienarbeit an der  
Technischen Universität Darmstadt  
Fachgebiet Theoretische Informatik  
Kryptographie und Computeralgebra  
Prof. Dr. Johannes Buchmann  
Alexander Wiesmaier

Lutz Feldgen, Ludwigsplatz 8 a, 64283 Darmstadt, [lfeldgen@gmx.net](mailto:lfeldgen@gmx.net)

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
<b>2</b>	<b>Analyse von Enterprise Application Servern</b>	<b>5</b>
2.1	Bea WebLogic Server 7.0 . . . . .	6
2.2	Borland Enterprise Server 5.1 . . . . .	6
2.3	IBM WebSphere 4.0 . . . . .	6
2.4	Macromedia JRun 4 . . . . .	6
2.5	Sun ONE Application Server Enterprise Edition 6.5/7.0 . . . . .	7
2.6	Java 2 Platform Enterprise Edition 1.3.1 . . . . .	7
2.7	JBoss 3.0 . . . . .	7
2.7.1	Container . . . . .	8
2.7.2	Clustering . . . . .	8
2.7.3	Transaktionen . . . . .	9
2.7.4	JBoss Container Managed Persistence (JBossCMP) . . . . .	10
2.7.5	Verbindung zur Datenbank . . . . .	10
2.7.6	Java Message Service (JMS) API . . . . .	10
2.7.7	Sonstiges . . . . .	11
<b>3</b>	<b>Analyse von Datenbanksystemen</b>	<b>12</b>
3.1	MySQL Server . . . . .	12
3.1.1	Lizensierung . . . . .	13
3.1.2	Performance . . . . .	13

<i>INHALTSVERZEICHNIS</i>	3
3.1.3 Clustering . . . . .	13
3.1.4 Datenbanken . . . . .	14
3.1.5 Sonstiges . . . . .	14
3.2 PostgreSQL . . . . .	14
3.2.1 Lizenzierung . . . . .	15
3.2.2 Die Query-Language . . . . .	15
3.2.3 Transactions und Concurrency . . . . .	15
3.2.4 Sonstiges . . . . .	15
3.2.5 Nachteile . . . . .	16
3.3 HSQLDB . . . . .	16
3.3.1 Betrieb im Server Modus (3 Varianten) . . . . .	16
3.3.2 In-Process Modus . . . . .	17
3.3.3 Nachteile . . . . .	18
<b>4 Fazit</b>	<b>19</b>
4.1 Entscheidung für JBoss 3.0 . . . . .	19
4.2 Entscheidung für MySQL Server . . . . .	20

# Kapitel 1

## Einleitung

Das Kryptographische Institut der Technischen Universität Darmstadt entwickelt ein Trustcenter. Ein Teil dieses Trustcenters ist die Certification Authority (CA), die für die Erstellung und Verwaltung von Zertifikaten zuständig ist und auf einem Enterprise-Application-Server (EAS) zum Einsatz kommen soll. Zur persistenten Sicherung der erstellten Zertifikate und Revokationen wird ein Datenbank-Management-System (DBMS) benötigt.

Im Rahmen zweier Diplomarbeiten von cand. inf. Markus Winkler und cand. inf. Lutz Feldgen wird die CA in Enterprise Java Beans (EJB) -Technologie designed und implementiert.

Ziel dieser Studienarbeit ist es, verschiedene EAS und DBMS auf Eignung für den Einsatz hinsichtlich der CA zu prüfen. wobei cand. inf. Markus Winkler parallel hierzu eine Studienarbeit mit dem Thema „Analyse der Enterprise-Java-Beans Spezifikation Version 2.0 als Basis für FlexiTRUST“ anfertigt und damit Ansätze zum Design der CA liefert.

Es sollen möglichst aktuelle Entwicklungstechniken zum Einsatz kommen, daher kommen nur EAS in Frage, die den Spezifikationen J2EE 1.3 und EJB 2.0 entsprechen. Desweiteren ist Clustering zwingend notwendig, um ein einfaches Load Balancing vornehmen zu können. Ein weiterer wichtiger Punkt sind die anfallenden Kosten für die Lizenzierung der Produkte. Hier sind kostenlose bzw. günstigere Produkte bei gleicher oder ähnlicher Qualifikation den teureren Produkten vorzuziehen.

## Kapitel 2

# Analyse von Enterprise Application Servern

Sun Microsystems stellt in ihrem Internetauftritt eine Liste von EAS bereit [SM02b], die auf Kompatibilität zur J2EE 1.3 Spezifikation getestet wurden. Da diese Liste zu viele EAS enthält, um auf alle angegebenen Server einzugehen, beschränkt sich diese Studienarbeit auf sechs der populärsten Produkte. Zusätzlich wurde der OpenSource-Application-Server JBoss 3.0 der JBoss Group in die Liste aufgenommen, der zwar nicht explizit getestet wurde, der J2EE 1.3 Spezifikation aber voll entspricht.

Stellt sich bei der Untersuchung eines Produkts heraus, daß es aus wichtigen Gründen nicht für den Einsatz geeignet ist, wird darauf verzichtet, auf weitere Eigenschaften einzugehen.

Untersucht werden folgende Produkte:

- Bea WebLogic Server 7.0 [BS02].
- Borland Enterprise Server 5.1 [Bor02a].
- IBM WebSphere 4.0 [IBM02a].
- Macromedia JRun 4 [Mac02a].
- Sun ONE Application Server 6.5/7.0 [SM02c, SM02d].
- Java 2 SDK Enterprise Edition, Version 1.3.1 [SM02a].
- JBoss 3.0 [Gro02a].

## 2.1 Bea WebLogic Server 7.0

Der Bea WebLogic Server 7.0 entspricht den wichtigsten Anforderungen, allerdings liegen die Lizenzkosten pro CPU bei € 26.290,00 exklusive 18% der Netto-Lizenzkosten für Support einer Testumgebung oder 21% für Support einer produktiven Umgebung.

Diese Lizenzkosten liegen weit außerhalb des Budgets für das Projekt, daher wird hier nicht weiter auf Bea WebLogic Server 7.0 eingegangen.

## 2.2 Borland Enterprise Server 5.1

Der Borland Enterprise Server 5.1 erfüllt nahezu alle Anforderungen, die Ausnahme bildet die ausschließliche Unterstützung der JDK 1.3.1 [Bor02b]. Die Einschränkung auf eine spezielle Version der JDK ist nicht praktikabel, da eine hierdurch eine späterer Weiterentwicklung kaum möglich ist oder zumindest erheblich erschwert wird. Auch wenn dies nicht entscheidend wäre, stellen die Lizenzkosten in Höhe von € 14.120,00 pro CPU einen sehr hohen finanziellen Aufwand dar. Aus diesen Gründen wird von einer weiteren Untersuchung des ansonsten interessanten Produkts abgesehen.

## 2.3 IBM WebSphere 4.0

IBM Websphere ist nicht für das Projekt geeignet, da nur Windows NT und Windows 2000 als Plattformen unterstützt werden, weiterhin können lediglich IBM JDKs eingesetzt werden. Die Palette der unterstützten DBMS beschränkt sich auf die kommerziellen Produkte Oracle (\$ 15.000,00 pro CPU), Microsoft SQL Server (€ 7004,08 Standard Edition, 1 Prozessor), Sybase (Sybase Adaptive Server Enterprise, \$ 3.995,00), IBM DB2 (\$ 995,00) und Informix (IBM Informix Dynamic Server 9.x, \$ 47.095,00 pro CPU) [IBM02b]. Auch die Lizenzkosten in Höhe von \$ 10.788,00 tragen dazu bei, von einer detaillierten Analyse abzusehen.

## 2.4 Macromedia JRun 4

Macromedia JRun 4 erfüllt nahezu alle Anforderungen und ist mit \$ 899,00 auch akzeptabel im Preis, allerdings werden nur die kommerziellen Datenbanken

Oracle (\$ 15.000,00 pro CPU), Microsoft SQL Server (€ 7004,08 Standard Edition, 1 Prozessor), Sybase (Sybase Adaptive Server Enterprise, \$ 3.995,00), IBM DB2 (\$ 995,00) und Informix (IBM Informix Dynamic Server 9.x, \$ 47.095,00 pro CPU) [Mac02b] unterstützt. Diese Preise sind nicht ausschließlich von den Herstellern bezogen, es wurden durchschnittliche Preise in Online-Shops ermittelt, wenn der Hersteller keine Preisangaben machte.

## 2.5 Sun ONE Application Server Enterprise Edition 6.5/7.0

SUN ONE Application Server ist nicht für das Projekt geeignet, die Beschränkung auf die kommerziellen DBMS Oracle, Microsoft SQL Server und Sybase ist nicht akzeptabel. Die Preise der DBMS sind in Kapitel 2.3 bereits aufgeführt.

Weitere Punkte, die einen Einsatz ausschließen, sind Lizenzkosten von \$ 19.995,00 pro CPU und die Einschränkung der Betriebssysteme auf Sun Solaris 2.6 oder 8, Windows NT oder Windows 2000, IBM AIX 4.3.x und HP-UX 11.x. [SM02c].

Kurz vor Fertigstellung dieser Studienarbeit erschien die neue Version 7.0 in den Editionen Platform, Standard und Enterprise. Die Platform-Edition unterstützt zwar J2SE 1.4 und ist frei von Lizenzgebühren, eine generelle Einschränkung der Sun ONE Application Server auf die kommerziellen Datenbanken schließt aber den Einsatz der Version 7.0 ebenfalls aus [SM02d].

## 2.6 Java 2 Platform Enterprise Edition 1.3.1

In Java 2 Platform Enterprise Edition 1.3.1 ist ein Application Server enthalten. Dieser ist aber eher eine Beispielimplementierung als für den echten Einsatz gedacht und daher nicht für die Verwendung in einem Projekt zu empfehlen.

## 2.7 JBoss 3.0

JBoss 3.0 ist ein OpenSource Application Server der JBoss Group, der in Java implementiert ist und vollständig auf J2EE basiert. Er ist konform zu den Spezifikationen J2EE 1.3 und EJB 2.0 und unterstützt das SUN JDK ab 1.3.1. Die Version 3.0 bringt einen Jetty Webserver mit, es ist aber auch möglich ein Package mit Apache Tomcat zu installieren.

Da JBoss 3.0 sämtliche Anforderungen erfüllt, werden einige Eigenschaften untersucht.

Der Großteil der Informationen wurde aus [Sch02] entnommen.

### 2.7.1 Container

Die ContainerFactory unterstützt Auto-Deployment, dadurch können geänderte Teile in die Applikation eingefügt werden, ohne daß der Container neu gestartet werden muß.

### 2.7.2 Clustering

Clustering ist das logische Gruppieren von zwei oder mehr Instanzen eines Servers, der auf einem oder mehreren Rechnern läuft. Jede Instanz des Servers im Cluster wird Knoten genannt. Der Cluster präsentiert sich als einzelner Server, die Verteilung des Servers auf mehrere Instanzen und Rechner ist für den Client transparent.

JBoss Clustering liefert folgende Features, durch die es möglich ist, große skalierbare und robuste J2EE-Applikationen zu entwickeln:

#### **Automatic discovery:**

Werden Knoten dem Cluster hinzugefügt, finden sich alle Knoten gegenseitig ohne zusätzliche Konfiguration.

#### **Cluster-weit replizierter JNDI-Baum:**

Jeder Knoten im Cluster hält den vollständigen JNDI-Baum, dadurch erhält man höhere Ausfallsicherheit, da keine zentrale Einheit für den JNDI-Baum zuständig ist.

#### **Fail-over und load-balancing für JNDI, RMI und alle EJB-Typen:**

Fail-over bedeutet, daß bei Ausfall eines Service oder Objekts in einem Knoten auf replizierte Services oder Objekte in einem anderen Knoten zurückgegriffen wird. Dies ist möglich bei allen Services und Objekten, für die load-balancing unterstützt wird. Bei JBoss sind dies der JNDI-Baum, die RMI-Registry und alle EJB-Typen, Stateful-Session-Beans können unterstützt werden, da ihr Status repliziert werden kann.

### **Farming:**

Unter Farming versteht man das automatische Hot-Deploy von allen Knoten im Cluster.

Führt ein Knoten ein Hot-Deploy aus, erhalten alle anderen Knoten die entsprechenden Daten um ihrerseits ein Hot-Deploy durchzuführen. Dadurch wird sichergestellt, daß sämtliche Knoten die gleiche Version des Clients ausführen.

## **2.7.3 Transaktionen**

### **Verteilte Transaktionen**

JBoss unterstützt verteilte Transaktionen, also Transaktionen, die sich über mehrere Datenquellen erstrecken. Dazu implementiert JBoss einen eigenen Transaktionsmanager. Dieser kann direkt über die Java Transaction API (JTA) angesteuert werden, wenn nicht der Container die Transaktionen verwalten soll.

### **Deadlock-Erkennungs-System**

Konkurrierende Zugriffe auf eine Datenbank können zu Problemen führen, insbesondere wenn Transaktionen verwendet werden. Eine Transaktion besteht aus mehreren einzelnen Operationen auf der Datenbank, die entweder komplett oder gar nicht ausgeführt werden dürfen. Wenn ein Prozess in die Datenbank schreiben soll, sperrt er die entsprechende Tabelle, dies bezeichnet man als locking. Es gibt verschiedene Arten von Locking-Strategien und locks, auf die nicht weiter eingegangen wird. Wenn zwei Prozesse auf gegenseitig gesperrte Tabellen zugreifen wollen, ist das Ergebnis bei bestimmten lock-Arten ein Dead-Lock: Jeder der Prozesse wartet auf die Freigabe des Anderen, die erst erfolgt, wenn die Transaktion beendet oder abgebrochen wird.

JBoss 3.0 beinhaltet ein Deadlock-Erkennungs-System, das eine Deadlock-Exception bei einem der blockierenden Prozesse wirft. Dessen Transaktion wird abgebrochen und bereits erfolgte Änderungen in der Datenbank werden rückgängig gemacht. Der Aufrufer des unterbrochenen Prozesses erhält die Deadlock-Exception und kann entscheiden, ob die Transaktion wiederholt werden oder ob er selbst eine Exception ausgeben soll. Der nicht unterbrochene Prozess hat nun Zugriff auf die gewünschte Tabelle und kann seine Transaktion fortführen.

### 2.7.4 JBoss Container Managed Persistence (JBossCMP)

JBossCMP ist eine zur EJB 2.0 CMP 2.0 Spezifikation konforme Persistence-Engine. Sie ist nach eigenen Angaben einer der besten Ansätze für Java-Management-Extensions (JMX)-basierte Persistenz bei Application-Server Technologie.

Über die DOCTYPE Deklaration im ejb-jar.xml Deployment Descriptor wird die Version des EJB-Jar-Archivs ermittelt. Für Version 2.0 benutzt JBoss die JBossCMP, andernfalls die ältere JAWS CMP 1.1 Persistence Engine.

JAWS steht für Just Another Web Storage und ist eine API zum Abbilden von EJB-Objekten in relationale Datenbanken.

Die JBossCMP enthält vordefinierte Mappings zur einfachen Einbindung von Datenbanken, darunter auch Oracle, SQLServer, DB2, Sybase, HSQLDB, PostgreSQL und MySQL.

Sie erlaubt multiple Datenquellen, läßt eine Tabellenerstellung zur Deploy-Zeit zu und bietet volle Unterstützung von Java-Objekten und EJB-Referenzen.

JBossCMP kann im automatisierten Modus laufen, dies bedeutet wenig Administrations-Overhead.

### 2.7.5 Verbindung zur Datenbank

Alle Verbindungen zu Datenbanken und anderen Resource-Managern werden in JBoss 3.0 vom Java-Connector-Architecture (JCA) Framework verwaltet.

JCA liefert ein durchdachtes Modell, in dem der Application-Server Transaktionen, Sicherheit und Pooling oder Resource-Management behandelt. Bisher existieren nur wenige Datenbanken, die JCA Adapter bereitstellen, herkömmliche JDBC-Datenbanktreiber werden über jca-jdbc-Wrapper benutzt.

### 2.7.6 Java Message Service (JMS) API

Über JMS können zwischen Applikationen (oder Teilen davon) asynchron Messages verschickt werden. Die Messages werden nicht direkt an den Empfänger verschickt, sondern an eine Queue (point to point) oder ein Topic (publish/subscribe) gesendet, daher ist es für den Absender nicht wichtig, ob der eigentliche Empfänger empfangsbereit ist. Der Message Service wird von Message-Oriented-Middleware (MOM) implementiert, die auch JMS-Provider genannt wird. Das standardisierte Interface zum JMS-Provider ist die JMS API.

JBoss enthält JBossMQ, einen JMS 1.02b konformen JMS Provider. Dieser implementiert die JMS Spezifikation vollständig. Er verfügt über einige zusätzliche Features wie die Application-Services-Facility (ASF) [Cof02], auf die im Online-Manual von JBoss 2.4+ verwiesen wird [Ant02]. Ein weiteres zusätzliches Feature ist der Resource Adapter für JMS, der als J2EE-Connector aufgebaut und für jeden nicht-asynchronen JMS-Task aus innerhalb einer EJB gedacht ist.

Im Hinblick auf die Austauschbarkeit von Komponenten sollten möglichst nur Features verwendet werden, die ein beliebiger anderer JMS Provider auch bietet, der dieser Spezifikation entspricht.

### 2.7.7 Sonstiges

JBoss unterliegt der LGPL (Lesser GNU Public Licence), die im Unterschied zur GPL sicherstellt, daß die ursprünglichen Rechte des Benutzers an der Software nicht durch Distributoren beschnitten werden, die die Software verändert haben.

Zur Installation ganzer JBoss-Farmen kann der sogenannte MicroKernel von JBoss auf eine Boot-Diskette gespeichert werden. Beim Start des Systems wird auch der MicroKernel von JBoss gestartet, der sich von einem zentralen Http-Server die Konfiguration und die benötigten Klassen holt.

## Kapitel 3

# Analyse von Datenbanksystemen

Der Einsatz eines DBMS wird für die Persistenz der Zertifikate und Revokationslisten benötigt. Weitere Verwendung findet die Datenbank bei der internen Verwaltung der Verarbeitungsschritte.

Geforderte Eigenschaften an das DBMS sind hohe Geschwindigkeit und Datensicherheit bzw. -integrität.

Analysiert werden die bekannten OpenSource Produkte MySQL Server [Com02a], PostgreSQL [Gro02b] und das im JBoss-Paket mitgelieferte HSQLDB [Mül02a]. Andere Datenbanksysteme mit vergleichbarem Preis/Leistungsverhältnis waren nicht verfügbar.

Die Analyse der DBMS ist eine Aufstellung der jeweiligen nennenswerten Vor- und Nachteile. Da hier alle Produkte den minimalen Anforderungen genügen, wird jedes Produkt untersucht.

### 3.1 MySQL Server

MySQL Server ist ein OpenSource DBMS der MySQL AB Company, das inzwischen in der Alpha-Version 4.0 vorliegt. Die Firma beschäftigt eigene Entwickler für das Projekt MySQL Server, die den Kern des Projekts selbst geschrieben haben, also vollständig mit dem Code des Servers vertraut sind. Die daraus resultierende Code-Konsistenz ist die Basis für die überragende Stabilität und Geschwindigkeit des Servers. Das Hauptaugenmerk bei der Entwicklung lag von

Beginn an bei der Performance. Features die diese einschränken werden nur implementiert, wenn der Nutzen groß genug und der Performanceverlust vertretbar ist. MySQL Server ist von Anfang an multi-threaded umgesetzt worden und da hierfür Kernel-Threads benutzt werden, ist der Einsatz von Mehr-Prozessor-Systemen problemlos und sinnvoll möglich. Als Query-Language wird eine erweiterte ANSI SQL-92 Syntax benutzt, ein erklärtes Ziel ist die vollständige Unterstützung von ANSI SQL-99.

Der Großteil der Informationen über MySQL Server stammen aus der Online-Dokumentation [Com02b].

### 3.1.1 Lizenzierung

MySQL Server unterliegt der GPL. Wenn er in einer kommerziellen Anwendung zum Einsatz kommen soll, kann eine kommerzielle Lizenz erworben werden. Die Lizenzkosten für einen Server liegen momentan bei € 230,-.

### 3.1.2 Performance

MySQL Server bietet eine sehr gute Performance. Diese wird durch sehr schnelle, thread-basierte Speicher-Allokierung und sehr schnelle B-Baum Tabellen mit Index-Kompression erreicht. Weiterhin wird ein Query-Cache eingesetzt, der die Geschwindigkeit für lesende Zugriffe um ein Vielfaches erhöht. Für weitere Geschwindigkeitssteigerung können read-only-Tabellen komprimiert werden.

### 3.1.3 Clustering

MySQL Server selbst unterstützt keine geclusterten Datenbanken, dafür wurden Replication Services implementiert. Diese Services setzen auf die Master / Slave Beziehung zwischen zwei oder mehr Datenbank-Servern auf, wobei ein Server als Master dient, die anderen als Slaves. Der Master hält ein binäres Log der Updates, die Slaves merken sich, wann sie das letzte Mal aktualisiert wurden und erhalten bei Verbindung zum Master ein entsprechendes Update. Dieses Verfahren ist nur bei überwiegenden Lesezugriffen sehr effektiv, da Updates nur über den Master erfolgen dürfen. Ein weiteres Anwendungsgebiet sind Backups. Um den Master des laufenden Systems nicht zu beeinträchtigen, kann einfach ein Datenbank-Backup eines Slave durchgeführt werden.

### 3.1.4 Datenbanken

MySQL Server unterstützt sehr große Datenbanken (es wird von einer Datenbank auf 15 Servern mit 60.000 Tabellen und insgesamt 5.000.000.000 Zeilen berichtet), zusätzlich können Queries auf Tabellen in verschiedenen Datenbanken erstellt werden. Zur Vermeidung von Fehlern bei INSERT müssen für alle Tabellenspalten Default-Werte definiert werden. Diese werden für jede Spalte eingesetzt, für die bei INSERT kein Wert angegeben wurde. Zur Indizierung sind bis zu 32 Indizes per Tabelle möglich, jeder Index kann aus 1 bis 16 Spalten oder Spaltenteilen bestehen.

Die Verbindung zur Datenbank wird über TCP Socket, Unix Sockets oder Named Pipes (NT) hergestellt, für Win32 gibt es ODBC-Unterstützung. Zur Administration der Datenbank wird "myisamchk" als schnelles Tool zur Verfügung gestellt, dessen Funktionalitäten aber auch über das SQL Interface verfügbar sind.

### 3.1.5 Sonstiges

Seit Version 4.0.1 kann MySQL Server als Embedded-Server betrieben werden. Hinzugekommen ist unter anderem das Character Set "latin\_de", welches Umlaute unterstützt und die Sortierreihenfolge gemäß den Einträgen im Telefonbuch ermöglicht. Desweiteren wird seit dieser Version auch UNION unterstützt, ein Feature auf das lange gewartet wurde.

Die aktuelle Version 4.0 Alpha wird als sehr stabil eingeschätzt, jedoch gibt es für professionelle Anwendungen eine Empfehlung für die letzte stabile Version 3.2.3.

Erwähnenswert ist der hochwertige Support durch die Entwickler selbst, außerdem existiert eine sehr gute Online-Dokumentation und ein lesenswertes User-Forum.

## 3.2 PostgreSQL

PostgreSQL ist ein OpenSource Produkt der PostgreSQL Global Development Group (GDG).

Das Hauptziel der PostgreSQL GDG ist, viele Features zu implementieren, auch wenn darunter die Performance leidet, jedoch bemüht man sich, die Geschwindigkeitsverluste möglichst gering zu halten.

Der Großteil der Informationen wurde der Online-Dokumentation entnommen [Gro02c].

### 3.2.1 Lizenzierung

PostgreSQL unterliegt der BSD License und ist für jede Art der Nutzung kostenlos. Die PostgreSQL GDG bietet einen kommerziellen Support an, der nach unterschiedlichen Leistungen gestaffelt ist. Diese Staffelung reicht von 30-tägigem Installation-Support für \$ 199,00 bis zu Platinum-Support für \$ 19.999,00 pro Jahr der telefonischen 24 Stunden-Support mit bis zu 3 Ansprechpartnern umfaßt.

### 3.2.2 Die Query-Language

PostgreSQL unterstützt den Kern von SQL-99, zusätzlich gibt es Erweiterungen von SQL-92.

Unterstützt werden zum Beispiel:

- Foreign Keys für referentielle Integrität.
- sämtliche SQL-99 Join-Typen.
- Views.
- Trigger vor und/oder nach Operationen auf Tabellen.

### 3.2.3 Transactions und Concurrency

PostgreSQL ist ACID-Konform (Atomicity, Consistency, Isolation, Durability)

Üblicherweise werden Tabellen bei Schreibzugriffen exklusiv gesperrt, um das Lesen von Daten zu verhindern, die gerade verändert werden (dirty-read) oder das Schreiben auf Daten zu verhindern, die gerade verändert werden (dirty-write). Das für PostgreSQL entwickelte MVCC (Multi-Version Concurrency Control) verhindert das Sperren von Tabellenzeilen durch Schreib- oder Lesezugriffe anderer Clients, indem es die auszuführenden Operationen in einem Cache speichert und bei Anfragen die gültigen Daten aus dem Cache zurückgibt.

### 3.2.4 Sonstiges

PostgreSQL liefert eine Prozess-per-User-Architektur die sicherstellt, daß jeder Request immer abgearbeitet werden kann.

Es wird Write Ahead Logging (WAL) unterstützt: Bevor eine Schreiboperation ausgeführt wird, schreibt PostgreSQL diese Operation in das Log, sodaß bei einem Systemfehler diese Operation nachträglich ausgeführt werden kann.

### 3.2.5 Nachteile

PostgreSQL erfordert ein regelmässiges VAKUUM, um von UPDATE und DELETE freigewordenen Persistenz-Speicher zurückzugewinnen. Dafür werden bis Version 7.1 die Tabellen gesperrt, was einem regelmäßigen kurzfristigen Totalausfall der Datenbank gleichzusetzen ist.

Bei PostgreSQL ab 7.2 sperrt VAKUUM nicht mehr die Tabellen, aber bei großen Tabellen sind erhebliche Performanceeinbußen zu erwarten. Das neue tabellensperrende VAKUUM FULL sollte öfter ausgeführt werden, wenn regelmäßig große Teile einer Tabelle gelöscht werden.

## 3.3 HSQLDB

Die HSQLDB Engine entstand als Open-Source-Projekt aus Hypersonic SQL, als dieses Projekt eingestellt wurde. HSQLDB ist eine kleine und schnelle Database Engine, die vollständig in Java implementiert ist. Zum Softwarepaket gehören einige Management Tools, die das Verwalten und Überprüfen der erstellten Datenbank erleichtern.

Eine Besonderheit ist, daß HSQLDB in verschiedenen Modi betrieben werden kann. Es gibt drei Server-Modi und einen In-Process-Modus, der den Betrieb in der Virtual-Machine der Applikation erlaubt.

Die Informationen zu HSQLDB wurden dem im Softwarepaket mitgeliefertem hsqlGuide und der Online angebotenen Feature-Liste entnommen [Mül02b].

### 3.3.1 Betrieb im Server Modus (3 Varianten)

Die Server-Modi bieten maximale Zugriffsmöglichkeit. Die Datenbank-Engine läuft in einer Java-Virtual-Machine (JVM) und lauscht auf Verbindungen von Programmen, die sowohl auf demselben Computer als auch auf Computern im Netzwerk laufen können. Es sind mehrere Zugriffe von verschiedenen Programmen gleichzeitig möglich, sie können sich mit dem Server über den HSQLDB-JDBC-Treiber verbinden. In jedem Server-Modus bietet der Server nur Verbindungen zu der Datenbank an, die zur Startzeit des Server spezifiziert wurde. Soll auf mehrere Datenbanken zugegriffen werden, sollte für jede Datenbank eine Server-Instanz gestartet werden, die auf einem eigenen Port lauscht.

Die Server-Modi unterscheiden sich durch das Protokoll, über das die Kommunikation zwischen Clients und Server läuft.

### **HDQLDB Server**

Dies ist die bevorzugte und performanteste Möglichkeit, einen Datenbank Server zu betreiben. Zur Kommunikation wird ein proprietäres Protokoll benutzt.

### **HSQLDB Web Server**

Als Web Server kann HSQLDB installiert werden, wenn Zugriffe von JDBC-Clients auf die Datenbank nur über das HTTP-Protokoll zulässig beziehungsweise möglich sind.

Da das HTTP-Protokoll zustandslos ist, existiert keine Session-Persistenz für Verbindungen zum Server. Jedes JDBC-Kommando wird als neue Verbindung zur Datenbank aufgefaßt, daher können keine Transaktionen rückgängig gemacht werden.

### **HSQLDB Servlet**

Die Installation als Servlet benutzt wie der Web Server das HTTP-Protokoll und hat daher die gleichen Einschränkungen. Sie wird verwendet, wenn ein eine separate Servlet-Engine oder ein separater Application-Server den Zugriff auf die Datenbank ermöglichen soll. Im Servletmodus kann HSQLDB nicht unabhängig von der Servlet Engine gestartet werden.

Da Web Server und Servlet kein Web-Frontend zur Datenbank beinhalten, können Clients nur über den JDBC-Treiber auf die Datenbank zugreifen.

### **3.3.2 In-Process Modus**

Dieser Modus betreibt die Datenbank-Engine als Teil der Applikation in der gleichen JVM. Für einige Applikationen ist dieser Modus schneller als die Server-Modi, da die Daten nicht konvertiert und über das Netzwerk versandt werden müssen. Der Hauptnachteil ist, daß nicht von außerhalb der Applikation auf die Datenbank zugegriffen werden kann. Dies bedeutet zusätzlich, daß die Daten nicht mit externen Werkzeugen überprüft werden können während die Applikation läuft.

Als Besonderheit ist es möglich, bei der Entwicklung zunächst einen der Server-Modi zu verwenden, um mit externen Werkzeugen auf die Datenbank zugreifen zu können. Wenn die Applikation dann eingesetzt werden soll, kann einfach auf den In-Process-Modus umgeschaltet werden.

Es ist außerdem möglich, die HSQLDB im In-Memory Modus zu betreiben. Hierbei existiert die Datenbank nur im Hauptspeicher, daher bietet dieser Modus keine Persistenz. Er sollte nur für interne Verarbeitung von Daten verwendet werden.

### 3.3.3 Nachteile

- Die Speicherung von Tabellen ist die einzige Art von Persistenz, die HSQLDB unterstützt. Eine eigene Partition mit eigenem Filesystem ist nicht möglich.
- Support von JDK 1.1.x, 1.2.x, SE 1.3.x, der Support von JDK SE 1.4.x3 wird in Aussicht gestellt.
- HAVING, ANY und ALL sind nicht verfügbar.
- Es gibt keine Trigger und Views (eventuell 1.7).
- Es ist nicht konform zu ACID.
- Es werden keine Server-Side Cursors unterstützt, daher gibt es Probleme mit grossen Ergebnissen, weil durch das In-Memory-Processing nur begrenzt Speicher zur Verfügung steht. Es existiert zwar ein Workaround, dieser macht die Queries aber unnötig kompliziert.
- Es wird nur ein Teil von ANSI-92 SQL unterstützt, eine genaue Auflistung ist unter [Mül02c] zu finden.

# Kapitel 4

## Fazit

### 4.1 Entscheidung für JBoss 3.0

#### **Bea WebLogic Server 7.0**

Grundsätzlich einsetzbar, aber die Lizenzkosten in Höhe von über € 26.290,00 sind unerschwinglich.

#### **Borland Enterprise Server 5.1**

Grundsätzlich einsetzbar, aber die Lizenzkosten in Höhe von über € 14.120,00 sind unerschwinglich.

#### **IBM WebSphere 4.0**

Es werden als Plattformen nur Windows NT und 2000, als DBMSen nur kommerzielle Produkte unterstützt.

#### **Macromedia JRun 4**

Die ausschließliche Unterstützung von kommerziellen DBMSen ist Macromedia JRun 4 nicht akzeptabel.

## **SUN ONE Application Server**

Auch hier werden nur kommerzielle DBMSe unterstützt, zusätzlich liegen die Lizenzkosten in Höhe von \$ 19.995,00 pro CPU außerhalb des finanziellen Rahmens.

## **Java 2 Platform Enterprise Edition**

Dies ist lediglich eine Beispielimplementierung und nicht für den Einsatz gedacht.

## **JBoss 3.0**

JBoss 3.0 genügt sämtlichen Anforderungen, ist kostenfrei und hat einen hervorragenden Support. Darüberhinaus bringt JBoss 3.0 einen Jetty Webserver mit, falls ein Webserver benötigt werden sollte. Da in früheren Versionen Apache Tomcat mitgeliefert wurde, ist JBoss 3.0 auch mit diesem Webserver erhältlich.

Aufgrund der ermittelten Vor- und Nachteile der einzelnen Enterprise-Application-Server wird vorgeschlagen, JBoss 3.0 einzusetzen.

## **4.2 Entscheidung für MySQL Server**

Grundsätzlich ist es schwierig, die betrachteten Datenbank-Management-Systeme miteinander zu vergleichen, da sie unterschiedliche Ziele verfolgen. MySQL Server zielt nahezu ausschließlich auf Performance ab, während PostgreSQL die Maximierung der Features bevorzugt und HSQLDB auf Kompaktheit und Ausgewogenheit setzt. Daß jede Ausrichtung auf ein bestimmtes Ziel Abstriche bei den übrigen Eigenschaften nach sich zieht, liegt auf der Hand. Sicherlich erfüllt jedes der Systeme die an es gestellten Anforderungen, da aber Performance und Verfügbarkeit die wichtigsten Anforderungen sind, wird der Einsatz von MySQL Server empfohlen.

# Abkürzungsverzeichnis

ANSI	American National Standards Institute
API	Application Programming Interface
ASF	Application Service Facility
BSD	Berkeley Software Design
CA	Certification Authority
CMP	Container Managed Persistence
DBMS	Database Management System
EAS	Enterprise Application Server
EJB	Enterprise JavaBeans
GPL	GNU Public License
HTTP	Hypertext Transfer Protocol
J2EE	Java 2 Platform, Enterprise Edition
J2SE	Java 2 Platform, Standard Edition
JAWS	Just Another Web Storage
JCA	Java Connector Architecture
JDBC	Java DataBase Connectivity
JDK	Java Developers Kit
JMS	Java Message Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JTA	Java Transaction API
JVM	Java Virtual Machine
LGPL	GNU Lesser Public License
MOM	Message Oriented Middleware
MVCC	Multi-Version Concurrency Control
PostgreSQL GDG	PostgreSQL Global Developer Group
RMI	Remote Method Invocation
SQL	Structured Query Language
WAL	Write Ahead Logging

# Literaturverzeichnis

- [Ant02] Kap. 6 JBoss and JMS In: ANTMAN, Peter: *JBoss 2.4+ Documentation*. <http://www.jboss.org/online-manual/HTML/ch06.html>, Juni 2002. – JBoss Online Manual
- [Bor02a] BORLAND. *Borland Enterprise Server*. <http://www.borland.com/besappserver/index.html>. 2002
- [Bor02b] BORLAND. *Borland Enterprise Server*. [http://www.borland.com/besappserver/pdf/besa51\\_sysreqs.pdf](http://www.borland.com/besappserver/pdf/besa51_sysreqs.pdf). 2002
- [BS02] BEA SYSTEMS, Inc. *BEA Weblogic Server*. <http://www.bea.com/products/weblogic/server/index.shtml>. 2002
- [Cof02] Kap. 8 Development Tools and IDE Integration In: COFFEY, John P.: *JBoss 2.4+ Documentation*. <http://www.jboss.org/online-manual/HTML/ch08.html>, Juni 2002. – JBoss Online Manual
- [Com02a] COMPANY, MySQL A. *MySQL*. <http://www.mysql.com>. 2002
- [Com02b] COMPANY, MySQL A. *MySQL Documentation*. <http://www.mysql.com/doc/en/index.html>. 2002
- [Gro02a] GROUP, JBoss. *JBoss Enterprise Application Server*. <http://www.jboss.org>. 2002
- [Gro02b] GROUP, PostgreSQL Global D. *PostgreSQL*. <http://www.de.postgresql.org>. 2002
- [Gro02c] GROUP, PostgreSQL Global D. *PostgreSQL*. <http://www.de.postgresql.org/idocs/>. 2002
- [IBM02a] IBM. *IBM Websphere*. <http://www-3.ibm.com/software/webservers/appserv/>. 2002

- [IBM02b] IBM. *IBM Websphere*. [http://www-3.ibm.com/software/webservers/appserv/doc/v40/prereqs/ae\\_v40%3.htm](http://www-3.ibm.com/software/webservers/appserv/doc/v40/prereqs/ae_v40%3.htm). 2002
- [Mac02a] MACROMEDIA. *Macromedia JRun 4*. <http://www.macromedia.com/software/jrun/>. 2002
- [Mac02b] MACROMEDIA. *Macromedia JRun 4*. [http://www.macromedia.com/software/jrun/productinfo/system\\_reqs/](http://www.macromedia.com/software/jrun/productinfo/system_reqs/). 2002
- [Mül02a] MÜLLER, Thomas. *HSQLDB*. <http://hsqldb.sourceforge.net/>. 2002
- [Mül02b] MÜLLER, Thomas. *HSQLDB Features*. <http://hsqldb.sourceforge.net/2/DocsandDev/features.html>. 2002
- [Mül02c] MÜLLER, Thomas. *HSQLDB Syntax*. <http://hsqldb.sourceforge.net/internet/hSqlSyntax.html>. 2002
- [Sch02] SCHÄFER, Andreas: *JBoss 3.0 Quick Start Guide*. <http://unc.dl.sourceforge.net/sourceforge/jboss/JBoss.3.0QuickStart.Dra%ft3.pdf>. 2002. – JBoss Quick Start
- [SM02a] SUN MICROSYSTEMS, Inc. *J2EE Application Server*. [http://java.sun.com/j2ee/sdk\\_1.3/](http://java.sun.com/j2ee/sdk_1.3/). 2002
- [SM02b] SUN MICROSYSTEMS, Inc. *List of EAS compatible to J2EE Spec 1.3*. <http://java.sun.com/j2ee/compatibility.html>. 2002
- [SM02c] SUN MICROSYSTEMS, Inc. *Sun ONE Application Server 6.5*. [http://www.sun.com/software/products/appsrvr\\_ee/home\\_appsrvr\\_ee6\\_5.htm%1](http://www.sun.com/software/products/appsrvr_ee/home_appsrvr_ee6_5.htm%1). 2002
- [SM02d] SUN MICROSYSTEMS, Inc. *Sun ONE Application Server 7.0*. [http://www.sun.com/software/products/appsrvr\\_pe/home\\_appsrvr\\_pe.html](http://www.sun.com/software/products/appsrvr_pe/home_appsrvr_pe.html). 2002